

The image displays a large, symmetrical, abstract shape composed of the letters 'S' and 'Y' arranged in a grid-like pattern. The shape is centered and has a complex, multi-lobed structure. The letters are arranged in a way that creates a sense of depth and three-dimensionality, with some letters appearing to be in the foreground and others in the background. The overall effect is a highly stylized, geometric representation of a letter or a complex figure.

```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIIIII          SSSSSSSS

```

04  
04  
04  
04  
04  
04  
04  
04  
04  
04

(2)	267	DECLARATIONS
(5)	752	EXESPROCSTRT - STARTUP NEW PROCESS
(6)	1303	EXIT IMAGE AND RUN DOWN FILES
(7)	1327	CATCH ALL CONDITION HANDLER
(8)	1407	EXESRMSEXH - EXEC Mode Exit Handler
(9)	1442	XQPMERGE - Merge the XQP into P1 Space
(10)	1534	IMAGE DUMP MERGE
(10)	1612	CRELNM - FIXUP AND INSERT A LOGICAL NAME BLOCK
(11)	1686	EXESCRE_JGTABLE - CREATE GROUP AND JOB-WIDE LOGICAL NAME TABLES

[illegible]

```

0000 1      .TITLE  PROCSTRT - PROCESS STARTUP AND INITIALIZATION
0000 2      .IDENT  'V04-002'
0000 3
0000 4
0000 5 *****
0000 6
0000 7      COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      ALL RIGHTS RESERVED.
0000 10
0000 11      THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12      ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13      INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14      COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15      OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16      TRANSFERRED.
0000 17
0000 18      THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19      AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20      CORPORATION.
0000 21
0000 22      DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23      SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25 *****
0000 26
0000 27
0000 28 ++
0000 29 FACILITY: EXECUTIVE, PROCESS CREATION SYSTEM SERVICE
0000 30
0000 31 ABSTRACT:
0000 32      PROCSTRT CONTAINS THE CODE NECESSARY TO CONCLUDE THE CREATION
0000 33      OF A PROCESS WHICH MUST BE EXECUTED IN THE CONTEXT OF THAT PROCESS.
0000 34
0000 35 ENVIRONMENT:
0000 36      MODE=KERNEL, EXECUTING IN CONTEXT OF NEW PROCESS
0000 37
0000 38 AUTHOR: R. I. HUSTVEDT      , CREATION DATE: 27-DEC-76
0000 39
0000 40 MODIFIED BY:
0000 41
0000 42      V03-002 RAS0332      Ron Schaefer      14-Sep-1984
0000 43      Check for RMSS_BUSY status in the RMS exit handler
0000 44      so as to prevent an infinite loop if the handler
0000 45      has interrupted RMS rundown badly. In that case,
0000 46      give up on trying to do rundown cleanly.
0000 47      Also, change the rundown type to do a full PPF rundown.
0000 48
0000 49      V04-001 JWT0195      Jim Teague      11-Sep-1984
0000 50      Replace RMS exec mode exit handler for proper rundown
0000 51      of single-image processes.
0000 52
0000 53      V03-043 LJK0288      Lawrence J. Kenah      9-Aug-1984
0000 54      The AUTHPRI cell now exists in both the PCB and the PHD.
0000 55
0000 56      V03-042 ACG0440      Andrew C. Goldstein,      24-Jul-1984 10:45
0000 57      Add ref count field to ORB

```

PR  
VO[illegible]

78

0000	58	:	
0000	59	:	
0000	60	:	V03-041 MH0040 Hai Huang 19-Jul-1984
0000	61	:	Add routine EXESCRE_GTABLE.
0000	62	:	
0000	63	:	V03-040 LMP0275 L. Mark Pilant, 12-Jul-1984 20:14
0000	64	:	Initialize the ACL info in the ORB to be a null descriptor
0000	65	:	list rather than an empty queue. This avoids the overhead
0000	66	:	of locking and unlocking the ACL mutex, only to find out
0000	67	:	that the ACL was empty.
0000	68	:	
0000	69	:	V03-039 RAS0319 Ron Schaefer 29-Jun-1984
0000	70	:	Initialize the logical name table name translation
0000	71	:	cache queue to empty.
0000	72	:	
0000	73	:	V03-038 LJK0272 Lawrence J. Kenah 10-Apr-1984
0000	74	:	Initialize VECSET array at the same time that the VECRESET
0000	75	:	array is set up.
0000	76	:	
0000	77	:	V03-037 MHB0121 Mark Bramhall 9-Apr-1984
0000	78	:	Move new spawn CLI information to P1 space.
0000	79	:	
0000	80	:	V03-036 TMK0010 Todd M. Katz 27-Mar-1984
0000	81	:	Modify the logical name system services to make use of the
0000	82	:	updated internal protection checking mechanisms. What this
0000	83	:	involves is replacing the CHIP protection template in the
0000	84	:	templates for the group and job-wide logical name table with
0000	85	:	a template for a quad-word aligned Object Rights Block, and
0000	86	:	making sure that the appropriate fields within the Object Rights
0000	87	:	blocks are initialized when group and job-wide logical name
0000	88	:	tables are created.
0000	89	:	
0000	90	:	V03-035 WMC0007 Wayne Cardoza 21-Mar-1984
0000	91	:	Create the default image I/O segment.
0000	92	:	
0000	93	:	V03-034 TMK0009 Todd M. Katz 07-Mar-1984
0000	94	:	Add a hash code field, LNM\$W_HASH, to every translation block
0000	95	:	of every logical name and logical name table template defined.
0000	96	:	This hash code field will be used in an optimization of logical
0000	97	:	name table name processing.
0000	98	:	
0000	99	:	V03-033 TMK0008 Todd M. Katz 17-Feb-1984
0000	100	:	Fix alignment problems with LNM\$GROUP and LNM\$JOB introduced
0000	101	:	by one of the two logical name table alignment bug fixes
0000	102	:	below. This is done by defining the symbols GROUP_XEND_SIZE
0000	103	:	and JOB_XEND_SIZE. These two symbols represent the actual
0000	104	:	amount of storage utilized by LNM\$GROUP and LNM\$JOB
0000	105	:	respectively while the two symbols GROUP_SIZE and JOB_SIZE
0000	106	:	represent the amount of storage actually allocated for these
0000	107	:	logical names. These two new symbols are needed by the PROCSTRT
0000	108	:	code that constructs the equivalence strings for these logical
0000	109	:	names. This code depends upon knowledge of the actual amount of
0000	110	:	storage, allocated to the logical names, which is utilized by
0000	111	:	the logical names.
0000	112	:	
0000	113	:	V03-032 LY00b8 Larry Yetto 17-FEB-1984 14:36
0000	114	:	Fix alignment of logical name tables

0000	115	:	V03-031	LY00b6	Larry Yetto	16-FEB-1984 14:21
0000	116	:		Fix alignment of logical name tables		
0000	117	:				
0000	118	:	V03-030	WMC0006	Wayne Cardoza	12-Jan-1983
0000	119	:		Create DZRO space for XQP, CRF no longer an option.		
0000	120	:				
0000	121	:	V03-029	TMK0007	Todd M. Katz	26-Jan-1984
0000	122	:		Fix a bug introduced in TMK0006. In EXESCRE_JGTABLE, if an		
0000	123	:		existing group table is found when an attempt is made to		
0000	124	:		create-if a new group table, then the paged pool for what would		
0000	125	:		have become a new group logical name table must be deleted. This		
0000	126	:		was not being done. This resulted in multiple group tables with		
0000	127	:		identical names, and had the further undesirable affect of		
0000	128	:		causing paged pool to disappear over time as more and more of		
0000	129	:		these duplicate group logical name tables were created as a		
0000	130	:		by-product of process creation.		
0000	131	:				
0000	132	:	V03-028	LJK0258	Lawrence J. Kenah	18-Jan-1984
0000	133	:		Fix bug introduced in LJK0257. Save R4 and R5 before they		
0000	134	:		are destroyed by a MOVCS instruction.		
0000	135	:				
0000	136	:	V03-027	LJK0257	Lawrence J. Kenah	28-Dec-1983
0000	137	:		Add support for longer text strings in the PQB. Fix error		
0000	138	:		paths. Add initialization code for P1 lookaside list.		
0000	139	:				
0000	140	:	V03-026	SHZ0001	Stephen H. Zalewski	27-Dec-1983
0000	141	:		Remove RMS executive mode exit handler.		
0000	142	:				
0000	143	:	V03-025	TMK0006	Todd M. Katz	10-Nov-1983
0000	144	:		Optimize the logical name and logical name table creations that		
0000	145	:		are required to be done at process creation time. This is done		
0000	146	:		by replacing all \$CRELNT and \$CRELNM system service calls with		
0000	147	:		the corresponding code that hand constructs the logical name		
0000	148	:		blocks and oversees their insertion into the overall logical		
0000	149	:		name structure. In addition, group logical name tables will no		
0000	150	:		longer be created for sub-processes. As in the case of the		
0000	151	:		job-wide logical name table, it will be assumed that the group		
0000	152	:		logical name table for a sub-process already exists.		
0000	153	:				
0000	154	:	V03-024	TMK0005	Todd M. Katz	12-Oct-1983
0000	155	:		If the process being created is not a sub-process, create the		
0000	156	:		job-wide logical name table.		
0000	157	:				
0000	158	:	V03-023	TMK0004	Todd M. Katz	26-Sep-1983
0000	159	:		Change the protection on the group logical name table to		
0000	160	:		SYSTEM:RWED OWNER: GROUP:R WORLD: so that processes with system		
0000	161	:		access rights can access and modify any group table.		
0000	162	:				
0000	163	:	V03-022	RAS0181	Ron Schaefer	5-Sep-1983
0000	164	:		Convert creation of SYSS\$INPUT, SYSS\$OUTPUT, SYSS\$ERROR,		
0000	165	:		SYSS\$DISK and TT logical names to use \$CRELNM.		
0000	166	:				
0000	167	:	V03-021	TMK0003	Todd M. Katz	22-Aug-1983
0000	168	:		Create the Group Logical Name Table with the protection mask		
0000	169	:		G:R as part of the changes being made to the logical name table		
0000	170	:		protection mechanism to provide for upwards compatibility		
0000	171	:		between V3 and V4. In addition, specify the GROUP and NO_ALIAS		

0000 172 :  
0000 173 :  
0000 174 :  
0000 175 :  
0000 176 :  
0000 177 :  
0000 178 :  
0000 179 :  
0000 180 :  
0000 181 :  
0000 182 :  
0000 183 :  
0000 184 :  
0000 185 :  
0000 186 :  
0000 187 :  
0000 188 :  
0000 189 :  
0000 190 :  
0000 191 :  
0000 192 :  
0000 193 :  
0000 194 :  
0000 195 :  
0000 196 :  
0000 197 :  
0000 198 :  
0000 199 :  
0000 200 :  
0000 201 :  
0000 202 :  
0000 203 :  
0000 204 :  
0000 205 :  
0000 206 :  
0000 207 :  
0000 208 :  
0000 209 :  
0000 210 :  
0000 211 :  
0000 212 :  
0000 213 :  
0000 214 :  
0000 215 :  
0000 216 :  
0000 217 :  
0000 218 :  
0000 219 :  
0000 220 :  
0000 221 :  
0000 222 :  
0000 223 :  
0000 224 :  
0000 225 :  
0000 226 :  
0000 227 :  
0000 228 :

attributes while creating the Group Logical Name Table so that the table can not be aliased, and so it will get marked specially as a Group Logical Name Table.

There is no need to perform any protection checking of process-private logical name tables; therefore, process-private logical name tables are no longer created with CHIP protection structures. Remove the CHIP protection structure from the process space logical name directory template as well as any fixing up of this CHIP which was being done as part of process creation.

V03-020 WMC0005 Wayne Cardoza 02-Jul-1983  
assorted performance improvements.

V03-019 RAS0176 Ron Schaefer 28-Jul-1983  
Fix group logical name table creation to be in octal;  
and clean up the code somewhat.

V03-018 LJK0221 Lawrence J. Kenah 5-Jul-1983  
Initialize listheads for image descriptor blocks.

V03-017 DMW4061 DMWalp 23-Jun-1983  
Change \$xxLNM value parameters to be by reference

V03-016 DMW4048 DMWalp 13-Jun-1983  
Fix protection problems with new logical name structures.  
Add execmode entry point to IMGDMF.

V03-015 ADE9005 Alan D. Eldridge 31-May-1983  
Make BSBW to MMG\$IMGRESET a JSB.

V03-014 RAS0158 Ron Schaefer 23-May-1983  
Add CHIP protection to logical name structures.  
Currently only SOGW protection is supported.  
Fix quota of LNM\$PROCESS\_DIRECTORY.

V03-103 WMC0003 Wayne Cardoza 10-May-1983  
Change XQP merge to use global sections rather than IMGACT.

V03-012 TMK0002 Todd M. Katz 26-Apr-1983  
Create the following logical name structures at process creation time:

1. LNM\$PROCESS\_TABLE.
2. LNM\$GROUP\_XXXXXXX (The Group Logical Name Table).
3. LNM\$GROUP.
4. LNM\$PROCESS.

Change the name of LNT\$PROCESS\_DIRECTORY to  
LNM\$PROCESS\_DIRECTORY.

V03-011 CDS0003 Christian D. Saether 20-Apr-1983  
Fix to V03-009. Don't merge xqp if EXESV\_INIT is clear.

V03-010 TMK0001 Todd M. Katz 14-Apr-1983  
Make the following changes to the setting up of the process

0000 229 :  
0000 230 :  
0000 231 :  
0000 232 :  
0000 233 :  
0000 234 :  
0000 235 :  
0000 236 :  
0000 237 :  
0000 238 :  
0000 239 :  
0000 240 :  
0000 241 :  
0000 242 :  
0000 243 :  
0000 244 :  
0000 245 :  
0000 246 :  
0000 247 :  
0000 248 :  
0000 249 :  
0000 250 :  
0000 251 :  
0000 252 :  
0000 253 :  
0000 254 :  
0000 255 :  
0000 256 :  
0000 257 :  
0000 258 :  
0000 259 :  
0000 260 :  
0000 261 :  
0000 262 :  
0000 263 :  
0000 264 :--

directory logical name table:

1. Make the table a kernel mode access table.
2. The address of the process directory table's table header is placed in LNMB\$SL\_TABLE.
3. The bit LNMT\$SV\_DIRECTORY is set in LNMT\$SB\_FLAGS.
4. The field LNMT\$SL\_LOGNAM is eliminated.

V03-009 CDS0002 Christian D. Saether 12-Apr-1983  
Always merge f11bxqp. Check for xqpmerge errors.

V03-008 WMC0002 Wayne Cardoza 01-Apr-1983  
Add second half of IMGDMF.

V03-007 WMC0001 Wayne Cardoza 14-Mar-1983  
Add image dump interface.

V03-006 ACG0305 Andrew C. Goldstein, 16-Dec-1982 14:03  
Get hibernate flag correctly in EXE\$PROCIMACT entry

V03-005 CDS0001 Christian D. Saether 16-Dec-1982  
Add routine to merge F11BXQP into P1 space.

V03-004 DMW4017 DMWalp 15-Dec-1982  
Added creation of new logical name hash table and  
logical name process directory

V03-003 JWH0117 Jeffrey W. Horn 01-Nov-1982  
Make the sizes of the RMS Process IO Segment and the  
Process Allocation Region controlable via SYSGEN  
parameters.

V03-002 KDM0002 Kathleen D. Morse 28-Jun-1982  
Add \$DYNDEF.

```
0000 267      .SBTTL  DECLARATIONS
0000 268      ::
0000 269      :: INCLUDE FILES:
0000 270      ::
0000 271
0000 272      $CCBDEF      ; CHANNEL CONTROL BLOCK DEFINITIONS
0000 273      $CHFDEF      ; CONDITION HANDLER DEFINITIONS
0000 274      $CLMSGDEF     ; COMMAND INTERPRETER STATUS CODES
0000 275      $DYNDEF      ; DYNAMIC STRUCTURE TYPE CODES
0000 276      $IACDEF      ; IMAGE ACTIVATION FLAGS
0000 277      $IHDEF       ; IMAGE HEADER DESCRIPTOR DEFINITIONS
0000 278      $IMGACTDEF    ; IMAGE ACTIVATOR ARGUMENTS
0000 279      $JIBDEF      ; DEFINE JIB OFFSETS
0000 280      $JPIDEF      ; JPI ITEM CODES
0000 281      $IMPDEF      ; RMS IMPURE AREA DEFINITIONS
0000 282      $LNMDDEF     ; LOGICAL NAME DEFINITIONS
0000 283      $LNMSTRDEF    ; LOGICAL NAME STRUCTURE DEFINITIONS
0000 284      $OPDEF       ; SYMBOLIC NAMES FOR INSTRUCTION OPCODES
0000 285      $ORBDEF      ; DEFINE OBJECT RIGHTS BLOCK OFFSETS
0000 286      $PCBDEF      ; DEFINE PCB OFFSETS
0000 287      $PHDDEF      ; DEFINE PROCESS HEADER
0000 288      $PQBDEF      ; DEFINE PROCESS QUOTA BLOCK OFFSETS
0000 289      $PRDEF       ; DEFINE PROCESSOR REGISTERS
0000 290      $PRTDEF      ; DEFINE PAGE PROTECTION VALUES
0000 291      $PRVDEF      ; PRIVILEGE BIT DEFINITIONS
0000 292      $PSLDEF      ; DEFINE PSL FIELDS
0000 293      $RMSDEF      ; DEFINE RMS ERROR STATUSES
0000 294      $SECDEF      ; SECTION FLAGS
0000 295      $SGNDEF      ; DEFINE SYSGEN CONSTANTS
0000 296      $SSDEF       ; DEFINE SYSTEM STATUS CODES
0000 297      $STSDEF      ; DEFINE STATUS CODE FIELDS
0000 298
0000 299      ::
0000 300      :: ASSUMPTIONS ABOUT THE STRUCTURE OF LOGICAL NAME AND OBJECT RIGHTS BLOCKS:
0000 301      ::
0000 302
0000 303      ASSUME  LNMB$$_FLINK,      EQ,  0
0000 304      ASSUME  LNMB$$_FLINK+4,   EQ,  LNMB$$_BLINK
0000 305      ASSUME  LNMB$$_BLINK+4,   EQ,  LNMB$$_SIZE
0000 306      ASSUME  LNMB$$_SIZE+2,     EQ,  LNMB$$_TYPE
0000 307      ASSUME  LNMB$$_TYPE+1,    EQ,  LNMB$$_ACMODE
0000 308      ASSUME  LNMB$$_ACMODE+1,   EQ,  LNMB$$_TABLE
0000 309      ASSUME  LNMB$$_TABLE+4,    EQ,  LNMB$$_FLAGS
0000 310      ASSUME  LNMB$$_FLAGS+1,   EQ,  LNMB$$_NAME
0000 311
0000 312      ASSUME  LNMX$$_FLAGS,      EQ,  0
0000 313      ASSUME  LNMX$$_FLAGS+1,   EQ,  LNMX$$_INDEX
0000 314      ASSUME  LNMX$$_INDEX+1,   EQ,  LNMX$$_HASH
0000 315      ASSUME  LNMX$$_HASH+2,     EQ,  LNMX$$_XLATION
0000 316
0000 317      ASSUME  LNMTH$$_FLAGS,      EQ,  0
0000 318      ASSUME  LNMTH$$_FLAGS+1,   EQ,  LNMTH$$_HASH
0000 319      ASSUME  LNMTH$$_HASH+4,     EQ,  LNMTH$$_ORB
0000 320      ASSUME  LNMTH$$_ORB+4,     EQ,  LNMTH$$_NAME
0000 321      ASSUME  LNMTH$$_NAME+4,    EQ,  LNMTH$$_PARENT
0000 322      ASSUME  LNMTH$$_PARENT+4,   EQ,  LNMTH$$_CHILD
0000 323      ASSUME  LNMTH$$_CHILD+4,    EQ,  LNMTH$$_SIBLING
```

```
0000 324      ASSUME  LNMTH$S_SIBLING+4, EQ,  LNMTH$S_QTABLE
0000 325      ASSUME  LNMTH$S_QTABLE+4,  EQ,  LNMTH$S_BYTESLM
0000 326      ASSUME  LNMTH$S_BYTESLM+4, EQ,  LNMTH$S_BYTES
0000 327
0000 328      ASSUME  ORB$S_OWNER,          EQ,  0
0000 329      ASSUME  ORB$S_OWNER+4,        EQ,  ORB$S_ACL_MUTEX
0000 330      ASSUME  ORB$S_ACL_MUTEX+4,    EQ,  ORB$S_SIZE
0000 331      ASSUME  ORB$S_SIZE+2,         EQ,  ORB$S_TYPE
0000 332      ASSUME  ORB$S_TYPE+1,         EQ,  ORB$S_FLAGS
0000 333      ASSUME  ORB$S_FLAGS+3,         EQ,  ORB$S_REFCOUNT
0000 334      ASSUME  ORB$S_REFCOUNT+2,     EQ,  ORB$S_MODE_PROT
0000 335      ASSUME  ORB$S_MODE_PROT+8,    EQ,  ORB$S_SYS_PROT
0000 336      ASSUME  ORB$S_SYS_PROT+4,    EQ,  ORB$S_OWN_PROT
0000 337      ASSUME  ORB$S_OWN_PROT+4,    EQ,  ORB$S_GRP_PROT
0000 338      ASSUME  ORB$S_GRP_PROT+4,    EQ,  ORB$S_WOR_PROT
0000 339      ASSUME  ORB$S_WOR_PROT+4,    EQ,  ORB$S_ACL_COUNT
0000 340      ASSUME  ORB$S_ACL_COUNT+4,   EQ,  ORB$S_ACL_DESC
0000 341      ASSUME  ORB$S_ACL_DESC+4,    EQ,  ORB$S_MIN_CLASS
0000 342      ASSUME  ORB$S_MIN_CLASS+ORB$S_MIN_CLASS, -
0000 343      EQ,  ORB$S_MAX_CLASS
0000 344      ASSUME  ORB$S_MAX_CLASS+ORB$S_MAX_CLASS, -
0000 345      EQ,  ORB$S_LENGTH
0000 346
0000 347      ::
0000 348      :: MACROS:
0000 349      ::
0000 350
0000 351      .MACRO  CRELMN,XLATION,XLATION_ATTR,LNMX,LNMB
0000 352      BSBW   CRELMN
0000 353      .WORD  <XLATION>
0000 354      .WORD  <XLATION_ATTR>
0000 355      .WORD  <LNMX>
0000 356      .WORD  <LNMB>
0000 357      .END    CRELMN
0000 358
0000 359      ::
0000 360      :: EQUATED SYMBOLS:
0000 361      ::
0000 362
00000000 0000 363  NTKVEC=0          ;OFFSET TO NEXT FREE KERNEL VECTOR
00000100 0000 364  NTEVEC=256       ;OFFSET TO NEXT FREE EXEC VECTOR
00000200 0000 365  NTRVEC=512      ;OFFSET TO NEXT FREE RUNDWN VECTOR
00000300 0000 366  NTMVEC=768      ;OFFSET TO NEXT MESSAGE VECTOR
```

```
0000 368
0000 369 ::
0000 370 :: OWN STORAGE:
0000 371 ::
0000 372 ::
0000 373 .PSECT YPROCSTRT,5 ; PAGED PSECT
0000 374
0000 375 EXESGQ_SYSDISK:: ; DESCRIPTOR FOR SYSSDISK
49 44 24 53 59 53 00000008'010E0000' 0000 376 .ASCID /SYSSDISK/
48 53 000E
0010 377 DEFDESC: ; DEFAULT IMAGE FILE NAME
45 58 45 2E 00000018'010E0000' 0010 378 .ASCID /.EXE/
001C 379
42 41 39 38 37 36 35 34 33 32 31 30 001C 380 CHARS: .ASCII /0123456789ABCDEF/ ; CHARS FOR OCTAL (HEX) -> ASCII CONVS
46 45 44 43 0028
002C 381
002C 382 ::
002C 383 :: CATCH ALL HANDLER FATAL CONDITION MESSAGE SUFFIX.
002C 384 ::
002C 385
66 20 74 69 78 65 20 65 67 61 6D 69 002C 386 SUFFIX: .ASCIZ /image exit forced./ ;
00 2E 64 65 63 72 6F 0038
003F 387
003F 388 ::
003F 389 :: STRINGS FOR IMAGE DUMP MERGE.
003F 390 ::
003F 391
49 4C 24 53 59 53 00000047'010E0000' 003F 392 DEFAULTNAMDESC:
45 58 45 2E 3A 59 52 41 52 42 003F 393 .ASCID /SYSS$LIBRARY:.EXE/
004D
0057 394 IMGDMPPNAM:
50 4D 44 47 4D 49 0000005F'010E0000' 0057 395 .ASCID /IMGDMP/
0065 396
0065 397 ::
0065 398 :: TEMPLATES FOR THE LOGICAL NAME TABLES AND NAMES CREATED WITHIN PROCSTRT.
0065 399 ::
0065 400
0065 401 .ALIGN QUAD
0068 402 PROC_DIR: ; LNMSPROCESS DIRECTORY TEMPLATE
0068 403 .LONG 0 ; FORWARD LINK
00000000 006C 404 .LONG 0 ; BACK LINK
0058' 0070 405 .WORD PROC_DIR SIZE ; SIZE OF STRUCTURE
40 0072 406 .BYTE DYN$C_LNM ; TYPE OF STRUCTURE
00 0073 407 .BYTE PSL$C_KERNEL ; KERNEL ACCESS MODE
00000000 0074 408 .LONG 0 ; CONTAINING TABLE HEADER ADDRESS
19 0078 409 .BYTE LNMB$M_NO_ALIAS!- ; NO ALIAS ALLOWED
0079 410 LNMB$M_TABLE!- ; THIS IS A TABLE
0079 411 LNMB$M_NODELETE ; ... THAT CANNOT BE DELETED
53 53 45 43 4F 52 50 24 4D 4E 4C 00' 0079 412 .ASCIZ "LNMSPROCESS_DIRECTORY" ; DIRECTORY NAME AS COUNTED STRING
59 52 4F 54 43 45 52 49 44 5F 0085
15 0079
008F 413
02 008F 414 .BYTE LNMX$M_TERMINAL ; TERMINAL TRANSLATION
82 0090 415 .BYTE LNMX$C_TABLE ; SPECIAL TABLE TRANSLATION INDEX
0000 0091 416 .WORD 0 ; TRANSLATION HASH CODE
25 0093 417 .BYTE LNMT$K_LENGTH ; SIZE OF TABLE HEADER BLOCK
0094 418
```

```
0000002C 0094 419 PROC_DIR_LNMTH = . - PROC_DIR
02 0094 420 .BYTE LNMTH$M_DIRECTORY
00000000 0095 421 .LONG 0
00000000 0099 422 .LONG 0
00000000 009D 423 .LONG 0
00000000 00A1 424 .LONG 0
00000000 00A5 425 .LONG 0
00000000 00A9 426 .LONG 0
00000000 00AD 427 .LONG 0
7FFFFFFF 00B1 428 .LONG ^X7FFFFFFF
7FFFFFFF 00B5 429 .LONG ^X7FFFFFFF
00B9 430
04 00B9 431 .BYTE LNM$M_XEND
00BA 432 .ALIGN QUAD
00000058 00C0 433 PROC_DIR_SIZE = . - PROC_DIR
00C0 434
00000058 00C0 435 PROC_TABLE = . - PROC_DIR
00000000 00C0 436 .LONG 0
00000000 00C4 437 .LONG 0
0050' 00C8 438 .WORD PROC_TABLE_SIZE
40 00CA 439 .BYTE DYN$C_LNM
00 00CB 440 .BYTE PSL$C_KERNEL
00000000 00CC 441 .LONG 0
09 00D0 442 .BYTE LNM$M_NO_ALIAS!-
00D1 443 .BYTE LNM$M_TABLE
00D1 444 .ASCII "LNM$PROCESS_TABLE"
00E3 445
02 00E3 446 .BYTE LNM$M_TERMINAL
82 00E4 447 .BYTE LNM$C_TABLE
0000 00E5 448 .WORD 0
25 00E7 449 .BYTE LNMTH$K_LENGTH
00E8 450
00000080 00E8 451 PROC_TABLE_LNMTH = . - PROC_DIR
00 00E8 452 .BYTE 0
00000000 00E9 453 .LONG 0
00000000 00ED 454 .LONG 0
00000000 00F1 455 .LONG 0
00000000 00F5 456 .LONG 0
00000000 00F9 457 .LONG 0
00000000 00FD 458 .LONG 0
00000000 0101 459 .LONG 0
00000000 0105 460 .LONG 0
00000000 0109 461 .LONG 0
010D 462
04 010D 463 .BYTE LNM$M_XEND
010E 464 .ALIGN QUAD
00000050 0110 465 PROC_TABLE_SIZE = . - PROC_DIR - PROC_TABLE
0110 466
0110 467
000000A8 0110 468 PROCESS = . - PROC_DIR
00000000 0110 469 .LONG 0
00000000 0114 470 .LONG 0
0038' 0118 471 .WORD PROCESS_SIZE
40 011A 472 .BYTE DYN$C_LNM
00 011B 473 .BYTE PSL$C_KERNEL
```

: TABLE IS FOR A DIRECTORY  
: ADDRESS OF HASH TABLE  
: ADDRESS OF OBJECT RIGHTS BLOCK  
: ADDRESS OF CONTAINING LNMB BLOCK  
: ADDRESS OF PARENT TABLE  
: ADDRESS OF CHILD TABLE  
: ADDRESS OF SIBLING TABLE  
: ADDRESS OF TABLE HOLDING QUOTA  
: INITIAL QUOTA ( POSITIVE INFINITY )  
: REMAINING QUOTA ( POSITIVE INFINITY )  
: LAST TRANSLATION  
: LNM\$PROCESS TABLE TEMPLATE  
: FORWARD LINK  
: BACK LINK  
: SIZE OF STRUCTURE  
: TYPE OF STRUCTURE  
: KERNEL ACCESS MODE  
: CONTAINING TABLE HEADER ADDRESS  
: NON-ALIASABLE  
: A TABLE  
: TABLE NAME AS COUNTED STRING  
: TERMINAL TRANSLATION  
: SPECIAL TABLE TRANSLATION INDEX  
: TRANSLATION HASH CODE  
: SIZE OF TABLE HEADER BLOCK  
: FLAGS BYTE  
: ADDRESS OF HASH TABLE  
: ADDRESS OF OBJECT RIGHTS BLOCK  
: ADDRESS OF CONTAINING LNMB BLOCK  
: ADDRESS OF PARENT TABLE  
: ADDRESS OF CHILD TABLE  
: ADDRESS OF SIBLING TABLE  
: ADDRESS OF TABLE HOLDING QUOTA  
: INITIAL QUOTA ( POOLED )  
: REMAINING QUOTA ( POOLED )  
: LAST TRANSLATION  
: LNM\$PROCESS TEMPLATE  
: FORWARD LINK  
: BACK LINK  
: SIZE OF STRUCTURE  
: TYPE OF STRUCTURE  
: KERNEL ACCESS MODE

53 53 45 43 4F 52 50 24 4D 4E 4C 00'  
45 4C 42 41 54 5F  
11

```
00000000 011C 474 .LONG 0 ; CONTAINING TABLE HEADER ADDRESS
53 53 45 43 4F 52 50 24 4D 4E 4C 00' 0120 475 .BYTE 0 ; FLAGS BYTE
0B 0121 476 .ASCII "LNMSPROCESS" ; LOGICAL NAME AS COUNTED STRING
02 012D 477 ;
00 012D 478 .BYTE LNMXSM_TERMINAL ; TERMINAL TRANSLATION
0000 012E 479 .BYTE 0 ; TRANSLATION INDEX IS 0
53 53 45 43 4F 52 50 24 4D 4E 4C 00' 012F 480 .WORD 0 ; TRANSLATION HASH CODE
45 4C 42 41 54 5F 0131 481 .ASCII "LNMSPROCESS_TABLE" ; TRANSLATION AS COUNTED STRING
11 013D
04 0143 482 ;
04 0143 483 .BYTE LNMXSM_XEND ; LAST TRANSLATION
00000038 0144 484 .ALIGN QUAD
0148 485 PROCESS_SIZE = . - PROC_DIR - PROCESS
0148 486
000000E0 0148 487 GROUP = . - PROC_DIR ; LNMSGROUP TEMPLATE
00000000 0148 488 .LONG 0 ; FORWARD LINK
00000000 014C 489 .LONG 0 ; BACK LINK
0038' 0150 490 .WORD GROUP_SIZE ; SIZE OF STRUCTURE
40 0152 491 .BYTE DYN$C_LNM ; TYPE OF STRUCTURE
00 0153 492 .BYTE PSL$C_KERNEL ; KERNEL ACCESS MODE
00000000 0154 493 .LONG 0 ; CONTAINING TABLE HEADER ADDRESS
00 0158 494 .BYTE 0 ; FLAGS BYTE
50 55 4F 52 47 24 4D 4E 4C 00' 0159 495 .ASCII "LNMSGROUP" ; LOGICAL NAME AS COUNTED STRING
09 0159
02 0163 496 ;
00 0163 497 .BYTE LNMXSM_TERMINAL ; TERMINAL TRANSLATION
0000 0164 498 .BYTE 0 ; TRANSLATION INDEX IS 0
78 5F 50 55 4F 52 47 24 4D 4E 4C 00' 0165 499 .WORD 0 ; TRANSLATION HASH CODE
78 78 78 78 78 10 0167 500 .ASCII "LNMSGROUP_XXXXXX" ; TRANSLATION AS COUNTED STRING
04 0178 501 ;
00000031 0178 502 .BYTE LNMXSM_XEND ; LAST TRANSLATION
0179 503 GROUP_XEND_SIZE = . - PROC_DIR - GROUP
0179 504 .ALIGN QUAD
00000038 0180 505 GROUP_SIZE = . - PROC_DIR - GROUP
0180 506
00000118 0180 507 JOB = . - PROC_DIR ; LNMSJOB TEMPLATE
00000000 0180 508 .LONG 0 ; FORWARD LINK
00000000 0184 509 .LONG 0 ; BACK LINK
0030' 0188 510 .WORD JOB_SIZE ; SIZE OF STRUCTURE
40 018A 511 .BYTE DYN$C_LNM ; TYPE OF STRUCTURE
00 018B 512 .BYTE PSL$C_KERNEL ; KERNEL ACCESS MODE
00000000 018C 513 .LONG 0 ; CONTAINING TABLE HEADER ADDRESS
00 0190 514 .BYTE 0 ; FLAGS BYTE
42 4F 4A 24 4D 4E 4C 00' 0191 515 .ASCII "LNMSJOB" ; LOGICAL NAME AS COUNTED STRING
07 0191
02 0199 516 ;
00 0199 517 .BYTE LNMXSM_TERMINAL ; TERMINAL TRANSLATION
0000 019A 518 .BYTE 0 ; TRANSLATION INDEX IS 0
78 78 78 5F 42 4F 4A 24 4D 4E 4C 00' 019B 519 .WORD 0 ; TRANSLATION HASH CODE
78 78 78 78 78 10 019D 520 .ASCII "LNMSJOB_XXXXXXXX" ; TRANSLATION AS COUNTED STRING
01A9
019D
01AE 521
```

[illegible]

[illegible]

```

582      .BYTE    LNMX$M_XEND                ; LAST TRANSLATION
583      .ALIGN   QUAD
584      SYS$OUTPUT_SIZE = . - PROC_DIR - SYS$OUTPUT
585
586      SYS$ERROR = . - PROC_DIR                ; SYS$ERROR TEMPLATE
587      .LONG     0                            ; FORWARD LINK
588      .LONG     0                            ; BACK LINK
589      .WORD     SYS$ERROR_SIZE                ; SIZE OF STRUCTURE
590      .BYTE     DYN$C_LNM                    ; TYPE OF STRUCTURE
591      .BYTE     PSL$C_EXEC                    ; EXECUTIVE ACCESS MODE
592      .LONG     0                            ; CONTAINING TABLE HEADER ADDRESS
593      .BYTE     0                            ; FLAGS BYTE
594      .ASCII    "SYS$ERROR"                  ; LOGICAL NAME AS COUNTED STRING
595
596
597      SYS$ERROR_LNMX = . - PROC_DIR
598      .BYTE     0                            ; TRANSLATION ATTRIBUTES
599      .BYTE     0                            ; TRANSLATION INDEX IS 0
600      .WORD     0                            ; TRANSLATION HASH CODE
601      .BYTE     LNMX$M_XEND[POB$$_ERROR]; WORST CASE TRANSLATION AS COUNTED STRING

```

[illegible]

	10	0771	639				
		0782	640	.BYTE	LNMXSM_TERMINAL	:	TERMINAL TRANSLATION
	02	0782	641	.BYTE	LNMXSC_TABLE	:	SPECIAL TABLE TRANSLATION INDEX
	82	0783	642	.WORD	0	:	TRANSLATION HASH CODE
	0000	0784	643	.BYTE	LNMTSHK_LENGTH	:	SIZE OF TABLE HEADER BLOCK
	25	0786	644				
		0787	645	GROUP_TABLE_LNMTM = . - GROUP_TABLE			
	00000027	0787	646	.BYTE	LNMTSHM_SHAREABLE!	:	TABLE IS SHAREABLE
	05	0788	647		LNMTSHM_GROUP	:	A GROUP TABLE
		0788	648	.LONG	0	:	ADDRESS OF HASH TABLE
	00000000	078C	649	.LONG	0	:	ADDRESS OF OBJECT RIGHTS BLOCK
	00000000	0790	650	.LONG	0	:	ADDRESS OF CONTAINING LNMB BLOCK
	00000000	0794	651	.ADDRESS	LNMTSHM_SYSTEM_DIR_LNMTM	:	ADDRESS OF PARENT TABLE
	00000000	0798	652	.LONG	0	:	ADDRESS OF CHILD TABLE
	00000000	079C	653	.LONG	0	:	ADDRESS OF SIBLING TABLE
	00000000	07A0	654	.ADDRESS	LNMTSHM_SYSTEM_DIR_LNMTM	:	ADDRESS OF TABLE HOLDING QUOTA
	00000000	07A4	655	.LONG	0	:	INITIAL QUOTA ( POOLED )
	00000000	07AB	656	.LONG	0	:	REMAINING QUOTA ( POOLED )
		07AC	657				
	04	07AC	658	.BYTE	LNMXSM_XEND	:	LAST TRANSLATION
		07AD	659				
		07AD	660	.ALIGN	QUAD		
	00000050	07B0	661	GROUP_TABLE_ORB = . - GROUP_TABLE			
	00000000	07B0	662	.LONG	0	:	GROUP NUMBER + 0 MEMBER NUMBER
	0000 FFFF	07B4	663	.WORD	-1, 0	:	INITIALIZED ACL MUTEX
	0070	07B8	664	.WORD	GROUP_TABLE_ORB_SIZE	:	SIZE OF OBJECT RIGHTS BLOCK
	49	07BA	665	.BYTE	DYN\$C_ORB	:	BLOCK TYPE
	00	07BB	666	.BYTE	0	:	NOTE NO ACL AS YET
		07BC	667	.LONG	0	:	ZERO RESERVED WORD & REF COUNT
	00000000	07C0	668	.QUAD	0	:	OBJECT DOES NOT HAVE AN ACCESS MODE
	00000000	07C8	669	.LONG	*X00000000	:	SYSTEM PROTECTION IS RWED
	0000000F	07CC	670	.LONG	*X0000000F	:	OWNER PROTECTION
	0000000E	07D0	671	.LONG	*X0000000E	:	GROUP PROTECTION IS R
	0000000F	07D4	672	.LONG	*X0000000F	:	WORLD PROTECTION
	00000000	07D8	673	.LONG	0, 0	:	NULL INITIAL ACL
	00'00'00'00'00'00'00'00'00'00'00'00'00'	07E0	674	.BYTE	0[ORB\$S_MIN_CLASS]	:	MINIMUM CLASSIFICATION MASK
	00'00'00'00'00'00'00'00'00'00'00'00'00'	07EC					
	00'00'00'00'00'00'00'00'00'00'00'00'00'	07F4	675	.BYTE	0[ORB\$S_MAX_CLASS]	:	MAXIMUM CLASSIFICATION MASK
	00'00'00'00'00'00'00'00'00'00'00'00'00'	0800					
		0808	676	.ALIGN	5		
	00000070	0820	677	GROUP_TABLE_ORB_SIZE = . - GROUP_TABLE - GROUP_TABLE_ORB			
	000000C0	0820	678	GROUP_TABLE_SIZE = . - GROUP_TABLE			
		0820	679				
	000000C0	0820	680	JOB_TABLE = . - GROUP_TABLE		:	LNMSJOB_XXXXXXXX TEMPLATE
	00000000	0820	681	.LONG	0	:	FORWARD LINK
	00000000	0824	682	.LONG	0	:	BACK LINK
	00C0	0828	683	.WORD	JOB_TABLE_SIZE	:	SIZE OF STRUCTURE
	40	082A	684	.BYTE	DYN\$C_LNM	:	TYPE OF STRUCTURE
	00	082B	685	.BYTE	PSL\$C_KERNEL	:	KERNEL ACCESS MODE
	00000000	082C	686	.ADDRESS	LNMTSHM_SYSTEM_DIR_LNMTM	:	CONTAINING TABLE HEADER ADDRESS
	09	0830	687	.BYTE	LNMB\$M_NO_ALIAS!	:	NON-ALIASABLE
		0831	688		LNMB\$M_TABLE	:	A TABLE
		0831	689	.ASCII	'LNMSJOB_XXXXXXXX''	:	TABLE NAME AS COUNTED STRING
		0831					
		0842	690				

[illegible]

PROCSTRT  
V04-002

J 10  
- PROCESS STARTUP AND INITIALIZATION  
DECLARATIONS

16-SEP-1984 01:00:43 VAX/VMS Macro V04-00  
14-SEP-1984 22:32:32 [SYS.SRC]PROCSTRT.MAR;3

Page 17  
(4)

```
08E0 746      <JPI END,4>-      ; GETJPI LIST TERMINATOR
08E0 747      <SCRATCHSIZE,0>,-  ; SIZE OF AREA ADDRESS OFF OF FP
08E0 748      >
0024      IMGACT_INADR:
002C      IMGACT_RETADR:
0034      HDRBUF:
0234      PROCPRIV:
023C      IMAGPRIV:
0244      PHD_FLAGS:
0248      JPI-PROC:
0254      JPI-IMAG:
0260      JPI-FLAG:
026C      JPI-END:
0270      SCRATCHSIZE:
08E0 749
```

```
08E0 752      .SBTTL EXES$PROCSTRT - STARTUP NEW PROCESS
08E0 753
08E0 754      :++
08E0 755      : FUNCTIONAL DESCRIPTION:
08E0 756      :
08E0 757      : CALLING SEQUENCE:
08E0 758      :     NONE
08E0 759      :
08E0 760      : INPUT PARAMETERS:
08E0 761      :     SCH$GL_CURPCB - POINTS TO PCB OF CURRENT PROCESS
08E0 762      :     PCB$Q_PQB - POINTER TO PROCESS QUOTA BLOCK
08E0 763      :
08E0 764      : IMPLICIT INPUTS:
08E0 765      :     IPL = IPL$ASTDEL
08E0 766      :
08E0 767      : OUTPUT PARAMETERS:
08E0 768      :     NONE
08E0 769      :
08E0 770      : IMPLICIT OUTPUTS:
08E0 771      :     LOGICAL NAMES ARE DEFINED FOR 'SYS$INPUT', 'SYS$OUTPUT', AND 'SYS$ERROR'
08E0 772      :     BASED ON THE STRINGS PASSED IN THE PROCESS QUOTA BLOCK.
08E0 773      :
08E0 774      : COMPLETION CODES:
08E0 775      :     NONE
08E0 776      :
08E0 777      : SIDE EFFECTS:
08E0 778      :     NONE
08E0 779      :
08E0 780      :--
08E0 781
08E0 782
08E0 783      :
08E0 784      : The PQB address must be stored before any instruction that can cause a page
08E0 785      : fault. If a page fault occurs and the process is put into a resource wait
08E0 786      : state, then the PQB address will be lost because the EFWM field, used to
08E0 787      : store the resource number, overlaps PCB$Q_PQB. This forces the first
08E0 788      : two instructions into a nonpaged program section.
08E0 789      :
0000 790
0000 791      .PSECT AEXENONPAGED
0000 792
0000 793 EXES$PROCSTRT::
0000 794      MOVL    SCH$GL_CURPCB,R4      : STARTUP NEW PROCESS
0000 795      MOVL    PCB$Q_PQB(R4),R6      : GET POINTER TO CURRENT PCB
0000 796      JMP     G^EXE_PROCSTRT      : GET POINTER TO PROCESS QUOTA BLOCK
0000 797      : CONTINUE IN PAGEABLE EXEC
0000 798
0000 799      .PSECT YYPROCSTRT
0000 800
0000 801 EXE_PROCSTRT:
0000 802
0000 803      N O T E :   THERE CAN BE NO I/O TO A PROCESS CHANNEL BETWEEN HERE
0000 804      :           AND THE END OF THE NEW CHANNEL CREATION CODE.
0000 805
0000 806      MOVL    G^MMG$GL_RMSBASE,G^CTL$GL_RMSBASE ; SET RMS DISPATCHER BASE
0000 807      MOVL    G^MMG$GL_CTLBASVA,G^CTL$GL_CTLBASVA ; SET CTL BASE ADDRESS
0000 808      : INITIALIZE THE DISPATCH VECTORS.
```

```
55 00000000'9F 9E 08F6 809 :  
    65 04 9A 08FD 810 :  
    0100 C5 04 9A 0900 811 :  
    0200 C5 04 9A 0905 812 :  
    0300 C5 04 9A 090A 813 :  
    04 A5 05 9A 090F 814 :  
    0104 C5 05 9A 0913 815 :  
    0204 C5 05 9A 0918 816 :  
    0304 C5 05 9A 091D 817 :  
00000000'GF 04 A5 9E 0922 818 :  
00000000'GF 0104 C5 9E 092A 819 :  
00000000'GF 0204 C5 9E 0933 820 :  
00000000'GF 0304 C5 9E 093C 821 :  
00000000'GF 54 DO 0945 822 :  
    00000000'9F DO 094C 823 :  
55 04 44 A6 00 E1 0953 824 :  
    36 A5 20 AB 0958 825 :  
    095C 826 :  
    095C 827 :  
    095C 828 :  
    095C 829 :  
    095C 830 :  
52 00000000'GF 9E 095C 831 10$: MOVAB G^CTL$GL_KRPFL,R2 : GET LISTHEAD ADDRESS  
51 00000000'GF 9E 0963 832 :  
    50 00 DO 096A 833 :  
    0E 15 096D 834 :  
    04 B2 61 0E 096F 835 20$: INSQUE (R1),@4(R2)  
51 00000000'8F CO 0973 836 :  
    F2 50 F5 097A 837 :  
    097D 838 30$: SOBGTR R0,20$  
    097D 839 :  
    097D 840 :  
    097D 841 :  
    0982 842 :  
50 51 00000000'EF DO 0987 843 :  
    00000000'EF C3 098E 844 :  
    51 50 D1 099A 845 :  
    03 15 099D 846 :  
    50 51 DO 099F 847 :  
    51 3C A6 3C 09A2 848 10$: MOVZWL PQB$ _WSEXTENT(R6),R1 :  
    52 30 A6 3C 09A6 849 :  
    53 34 A6 3C 09AA 850 :  
    52 51 D1 09AE 851 :  
    03 18 09B1 852 :  
    51 52 DO 09B3 853 :  
    50 51 D1 09B6 854 20$: MOVZWL PQB$ _WSQUOTA(R6),R2 :  
    03 15 09B9 855 :  
    51 50 DO 09BB 856 :  
    51 52 D1 09BE 857 30$: MOVZWL PQB$ _WSDEFAULT(R6),R3 :  
    03 15 09C1 858 :  
    52 51 DO 09C3 859 :  
    52 53 D1 09C6 860 40$: CMPL R1,R2 :  
    03 15 09C9 861 :  
    53 52 DO 09CB 862 :  
50 08 A5 01 A3 09CE 863 50$: SUBW3 #1,PHD$W_WSLIST(R5),R0 :  
    51 50 A0 09D3 864 :  
    16 A5 51 B0 09D6 865 :  
    MOVW R1,PHD$W_WSEXTENT(R5) :  
: SET EXTENT  
: GET ADR OF 1ST VECTOR PAGE  
: SET OFFSET TO 1ST FREE KERNEL VECTOR  
: SET OFFSET TO 1ST FREE EXEC VECTOR  
: SET OFFSET TO 1ST FREE RUNDWN VECTOR  
: SET OFFSET TO 1ST FREE MESSAGE VECTOR  
: SET AN 'RSB' INTO THE 1ST FREE VECTOR  
: SET AN 'RSB' INTO THE 1ST FREE VECTOR  
: SET AN 'RSB' INTO THE 1ST FREE VECTOR  
: SET AN 'RSB' INTO THE 1ST FREE VECTOR  
: SET AN 'RSB' INTO THE 1ST FREE VECTOR  
: SET POINTER TO START OF VECTOR  
: SET POINTER TO START OF VECTOR  
: SET POINTER TO START OF VECTOR  
: SET POINTER TO START OF VECTOR  
: SET POINTER TO PCB  
: GET SAFE POINTER TO PROCESS HEADER WINDOW  
: IMAGE DUMP WAS REQUESTED  
: SET UP P1 SPACE LOOKASIDE LIST FOR KERNEL MODE BUFFERS  
: GET LISTHEAD ADDRESS  
: GET  
: GET MAXIMUM WORKING SET LIST LENGTH  
: GET AVAILABLE PAGES  
: MINIMIZE WITH SPECIFIED QUOTA  
: USE QUOTA  
: USE MAXIMUM WORKING SET COUNT  
: GET MAXIMUM PAGES FOR WORKING SET  
: GET MAXIMUM QUOTA FOR WORKING SET  
: GET DESIRED DEFAULT  
: EXTENT MUST BE BIGGER THAN QUOTA  
: YES, USE IT AS IS  
: FORCE TO QUOTA (EXTENT MAY BE 0)  
: EXTENT MUST BE LESS THAN MAX PAGES  
: BRANCH IF OK AS IS  
: SET EXTENT TO MAX MEMORY  
: QUOTA MUST BE LESS THAN EXTENT  
: BRANCH IF OK AS IS  
: SET QUOTA TO EXTENT  
: DEFAULT MUST BE LESS THAN QUOTA  
: BRANCH IF OK AS IS  
: SET DEFAULT TO QUOTA  
: GET BASE OFFSET TO WORKING SET LIST  
: GET EXTENT  
: SET EXTENT
```

```
14 A5 51 B0 09DA 866      MOVW    R1,PHDSW_WSAUTHEXT(R5)  ; SET AUTHORIZED EXTENT
      52 50 A0 09DE 867      ADDW    R0,R2                ; GET QUOTA
18 A5 52 B0 09E1 868      MOVW    R2,PHDSW_WSQUOTA(R5)  ; QUOTA VALUE
      0A A5 52 B0 09E5 869      MOVW    R2,PHDSW_WSAUTH(R5)  ; AUTHORIZED VALUE
1A A5 53 50 A1 09E9 870      ADDW3   R0,R3,PHDSW_DFWSCNT(R5) ; SAVE DEFAULT WORKING SET SIZE
      09EE 871
      09EE 872 : THE AUTHPRI CELL EXISTS IN TWO PLACES. THE $SETPRI SYSTEM SERVICE USES
      09EE 873 : THE PCB CELL BUT THE PHD CELL MUST EXIST FOREVER BECAUSE THAT IS WHERE
      09EE 874 : THE JPI ITEM CODE BELIEVES THAT AUTHPRI IS LOCATED.
      09EE 875
2B A4 2F A4 90 09EE 876      MOVW    PCB$B_Prib(R4),PCB$B_AUTHPRI(R4)  ; SET INITIAL PROCESS PRIORITY
010C C5 2F A4 90 09F3 877      MOVW    PCB$B_Prib(R4),PHD$B_AUTHPRI(R5)  ; ... IN BOTH PCB AND PHD
      6C B4 66 7D 09F9 878      MOVQ    PQB$Q_Prvmsk(R6),@PCB$B_Phd(R4)  ; SET PRIVILEGES FOR PROCESS
00000000'9F 66 7D 09FD 879      MOVQ    PQB$Q_Prvmsk(R6),@CTL$GQ_PROCPRIV  ; BOTH PERMANENT AND CURRENT
      00E0 C5 66 7D 0A04 880      MOVQ    PQB$Q_Prvmsk(R6),PHD$Q_AUTHPRIV(R5)  ; AND AUTHORIZED MASKS
00000000'9F 46 A6 90 0A09 881      MOVW    PQB$B_Msgmask(R6),@CTL$GB_Msgmask  ; GET DEFAULT MESSAGE FLAGS
00000000'9F 00000000'EF 7D 0A11 882      MOVQ    EXESGQ_SYSTIME,@CTL$GQ_LOGIN  ; SAVE LOGIN TIME
      7E 54 7D 0A1C 883      MOVQ    R4,-(SP)                ; SAVE PCB AND PHD POINTERS
      0A1F 884
      0A1F 885 : MOVE MINIMUM AND MAXIMUM AUTHORIZED SECURITY CLEARANCE RECORDS INTO THE PHD.
      0A1F 886 : THE FOLLOWING ASSUME STATEMENTS GUARANTEE THAT WE CAN SAFELY PERFORM THIS
      0A1F 887 : WITH A SINGLE MOVCL INSTRUCTION.
      0A1F 888
      0A1F 889      ASSUME PQB$S_MIN_CLASS EQ PHD$S_MIN_CLASS
      0A1F 890      ASSUME PQB$S_MAX_CLASS EQ PHD$S_MAX_CLASS
      0A1F 891      ASSUME PQB$R_MAX_CLASS EQ <PQB$R_MIN_CLASS + PQB$S_MIN_CLASS>
      0A1F 892      ASSUME PHD$R_MAX_CLASS EQ <PHD$R_MIN_CLASS + PHD$S_MIN_CLASS>
      0A1F 893
      0A1F 894      MOVCL    #<PQB$S_MIN_CLASS+PQB$S_MAX_CLASS>,-
      0A21 895      PQB$R_MIN_CLASS(R6),-
      0A23 896      PHD$R_MIN_CLASS(R5)
      0A26 897
      0A26 898 : INITIALIZE LISTHEADS FOR DOUBLY LINKED LISTS USED BY IMAGE ACTIVATOR
      0A26 899
50 00000000'9F 9E 0A26 900      MOVAB   @#IAC$GL_IMAGE_LIST,R0  ; LIST OF ACTIVATED IMAGES
      60 50 D0 0A2D 901      MOVL     R0,(R0)                ; INITIALIZE FLINK
      04 A0 50 D0 0A30 902      MOVL     R0,4(R0)              ; ... AND BLINK
      0A34 903
50 00000000'9F 9E 0A34 904      MOVAB   @#IAC$GL_WORK_LIST,R0  ; LIST OF WORK IN PROGRESS
      60 50 D0 0A3B 905      MOVL     R0,(R0)                ; INITIALIZE FLINK
      04 A0 50 D0 0A3E 906      MOVL     R0,4(R0)              ; ... AND BLINK
      0A42 907
50 00000000'9F 9E 0A42 908      MOVAB   @#IAC$GL_ICBFL,R0      ; ADDRESS OF ICB LOOKASIDE LIST
      60 50 D0 0A49 909      MOVL     R0,(R0)                ; INITIALIZE FLINK
      04 A0 50 D0 0A4C 910      MOVL     R0,4(R0)              ; ... AND BLINK
      0A50 911
      0A50 912
      0A50 913 : CREATE THE PAGES FOR THE CCB TABLE, PROCESS ALLOCATION REGION, AND DEFAULT
      0A50 914 : IMAGE I/O SEGMENT
      0A50 915
      0A50 916
53 00000000'GF 3C 0A50 917      MOVZWL  G*SGN$GW_PCHANCNT,R3  ; PICK UP SYSGEN PARAM FOR # CHANS
      53 53 D6 0A57 918      INCL     R3                    ; ALLOW FOR WASTED CCB
      53 10 C4 0A59 919      MULL     #CCB$C_LENGTH,R3      ; CONVERT TO # BYTES
53 000001FF 8F C0 0A5C 920      ADDL     #511,R3              ; ROUND UP TO EVEN PAGES
53 000001FF 8F CA 0A63 921      BICL     #511,R3
54 00000000'GF 3C 0A6A 922      MOVZWL  G*SGN$GW_CTLPAGES,R4  ; GET # PAGES FOR PROCESS ALL REGION
```

```
54 54 09 78 0A71 923 ASHL #9,R4,R4 : CONVERT TO # BYTES
53 54 C0 0A75 924 ADDL R4,R3 : GET TOTAL # BYTES NEEDED SO FAR
57 00000000'GF 3C 0A78 925 MOVZWL G*SGN$GW_PIO$PAGES,R7 : GET # PAGES FOR PIO SEGMENT
57 57 09 78 0A7F 926 ASHL #9,R7,R7 : CONVERT TO NUMBER OF BYTES
53 57 C0 0A83 927 ADDL R7,R3 : GET TOTAL # BYTES NEEDED
58 00000000'GF 3C 0A86 928 MOVZWL G*SGN$GW_IMGIOCNT,R8 : GET # PAGES FOR IIO SEGMENT
58 58 09 78 0A8D 929 ASHL #9,R8,R8 : CONVERT TO NUMBER OF BYTES
53 58 C0 0A91 930 ADDL R8,R3 : GET TOTAL # BYTES NEEDED
55 00000000'EF DE 0A94 931 MOVAL CTL$GL_CTLBASVA,R5 : GET POINTER TO 'TOP' OF P1
7E 65 01 C3 0A9B 932 SUBL3 #1,(R5),-(SP) : 'LAST' PAGE IN P1
7E 65 53 C3 0A9F 933 SUBL3 R3,(R5),-(SP) : 'TOP' OF CREATED REGION
52 7E 7E 0AA3 934 MOVAQ -(SP),R2 : SPACE FOR RETADR
00000D00 8F DD 0AA6 935 PUSHL #PSL$C_KERNEL+<PRT$C_UREW> : ACCESS MODE AND PROTECTION
52 DD 0AAC 936 PUSHL R2 : RETADR ARRAY
08 A2 9F 0AAE 937 PUSHAB 8(R2) : INADR ARRAY
03 DD 0AB1 938 PUSHL #3 : ARGUMENT COUNT
50 5E D0 0AB3 939 MOVL SP,R0
0AB6 940 $CMKRNL S - : CALL INTERNAL ENTRY POINT FOR $CRETVA
0AB6 941 ROUTIN = G*MMG$CRETVA,-
0AB6 942 ARGST = (R0)
08 A2 OD 50 E9 0AC5 943 BLBC R0,VABUG : GET OUT ON ERROR
07 D1 0AC8 944 CMPL (R2),8(R2) : DID WE GET FULL REQUEST?
0C A2 04 A2 D1 0ACC 945 BNEQ VABUG : NO, ERROR OUT
0D 13 0AD3 946 CMPL 4(R2),12(R2) : MAKE DOUBLY SURE
0AD5 947 BEQL DIVR : NO, ERROR OUT
00000000'FF 66 0E 0AD5 948 VABUG: INSQUE (R6),@EXES$GL_PQBBL : DEALLOCATE PQB TO LOOKASIDE LIST
0391 31 0ADC 950 SETIPL #0 : ALLOW PROCESS TO BE DELETED
0AE2 951 BRW EXES$EXIT_IMAGE : DELETE THE PROCESS
0AE2 952 :
0AE2 953 : NOW DIVIDE THE CREATED SPACE INTO FOUR AREAS
0AE2 954 :
0AE2 955 :
00000004'EF 62 D0 0AE2 956 DIVR: MOVL (R2),PIO$GQ_IIO$DEFAULT+4 : DEFAULT IMAGE I/O AREA
00000000'EF 58 D0 0AE9 957 MOVL R8,PIO$GQ_IIO$DEFAULT : SIZE
50 00000004'9F DE 0AF0 958 MOVAL @#PIO$GW_PIO$IMPA+IMP$C_IIO$SEGADDR,R0 : GET POINTER ADDRESS
58 62 C0 0AF7 959 ADDL (R2),R8 : START OF REMAINING SPACE
80 58 D0 0AFA 960 MOVL R8,(R0)+ : SET UP THE PIO SEG ADDR
60 57 D0 0AFD 961 MOVL R7,(R0) : SET LENGTH
50 57 58 C1 0B00 962 ADDL3 R8,R7,R0 : GET POINTER TO FREE SPACE
00000000'9F 50 D0 0B04 963 MOVL R0,@#CTL$GQ_ALLOCREG : SET UP PROCESS ALLOCATION
80 D4 0B08 964 CLRL (R0)+ : NULL FORWARD POINTER
60 54 D0 0B0D 965 MOVL R4,(R0) : SET SIZE OF REGION
50 00000000'GF 3C 0B10 966 MOVZWL G*SGN$GW_CTL$IMGLIM,R0 : GET IMAGE LIMIT
00000000'9F 50 09 78 0B17 967 ASHL #9,R0,@#CTL$GL_PRCALLCNT : CONVERT TO # BYTES
00000000'9F 04 A2 0F C3 0B1F 968 SUBL3 #CCB$C_LENGTH-T,4(R2),@#CTL$GL_CCB$BASE : STORE BASE OF CHANNEL TABL
00000000'9F 00000000'GF 3C 0B28 969 MOVZWL G*SGN$GW_PCHANCNT,@#CTL$GW_NMIOCH : SET NUMBER OF CHANNELS
0B33 970
0B33 971
0B33 972 :
0B33 973 : NOTE(!!!!): THE ABOVE ASSIGNMENT MUST BE DONE AT THE VERY END OF THIS
0B33 974 : SECTION OF CODE, AS THE CELL NMIOCH BEING NON-ZERO IS AN
0B33 975 : INDICATOR TO IOCS$FFCHAN THAT THERE IS ACTUALLY A REAL
0B33 976 : CHANNEL TABLE TO LOOK AT.
0B33 977 :
0B33 978 :
65 62 D0 0B33 979 MOVL (R2),(R5) : UPDATE BASE OF VA IN CTL REGION
```

```
SE 20 AE DE OB36 980      MOVAL 32(SP),SP      ; POP $CRETVA ARGS
OB3A 981
OB3A 982
OB3A 983
OB3A 984
OB3A 985
OB3A 986
OB3A 987
OB3A 988
OB3A 989
OB3A 990
51 00000000'GF D0 OB3A 991      MOVL G^LNMSGL HTBLSIZP,R1      ; RETRIEVE NUMBER OF HASH TABLE ENTRIES
51 0000000C 9F41 DE OB41 992      MOVAL @#LNMSH$K_BUCKET[R1],R1 ; MULTIPLY BY 4 AND ADD OVERHEAD
57 51 D0 OB49 993      MOVL R1,R7      ; SAVE SIZE OF HASH TABLE
51 000006F0 8F C0 OB4C 994      ADDL2 #P1_ALLOC_SIZE,R1      ; ADD IN SIZE OF LOGICAL NAME BLOCKS
00000000'GF 16 OB53 995      JSB G^EXESALOP1PROC      ; ALLOCATE TOTAL AMOUNT OF SPACE NEEDED
58 52 D0 OB59 996      MOVL R2,R8      ; SAVE ADDRESS OF ALLOCATED SPACE
OB5C 997
OB5C 998      MOVCS #P1_ALLOC_SIZE,-      ; COPY TEMPLATE FOR ALL LOGICAL NAMES
62 51 00 F505 CF OB60 999      PROC DIR,#0,R1,(R2)      ; AND ZERO PROCESS-PRIVATE HASH TABLE
53 58 000006F0 8F C1 OB66 1000      ADDL3 #P1_ALLOC_SIZE,R8,R3      ; COMPUTE HASH TABLE ADDRESS
OB6E 1001
50 00000000'9F 53 D0 OB6E 1002      MOVL R3,@#CTL$GL_LNMHASH      ; STORE ADDRESS OF HASH TABLE AWAY
00000000'GF 01 C3 OB75 1003      SUBL3 #1,G^LNMSGL_HTBLSIZP,R0 ; CALCULATE UPPER BOUND OF HASH INDEX
63 50 D2 OB7D 1004      MCOML R0,LNMHSH$L_MASK(R3)      ; STORE HASH INDEX MASK IN HASH TABLE
OB A3 57 B0 OB80 1005      MOVW R7,LNMHSH$W_SIZE(R3)      ; STORE HASH TABLE SIZE IN HEADER
38 90 OB84 1006      MOVB #DYN$C_RSHT,-      ; STORE HASH TABLE STRUCTURE TYPE IN
OA A3 OB86 1007      LNMHSH$B_TYPE(R3)      ; HASH TABLE HEADER
OB88 1008
OB88 1009
OB88 1010
OB88 1011
OB88 1012
OB88 1013
OB88 1014
57 2C A8 9E OB88 1015      MOVAB PROC DIR_LNMTH(R8),R7      ; COMPUTE DIRECTORY'S TABLE HEADER ADDR
01 A7 53 D0 OB8C 1016      MOVL R3,LNMTH$L_HASH(R7)      ; STORE HASH TABLE ADDR IN TABLE HEADER
0C A8 57 D0 OB90 1017      MOVL R7,LNMB$L_TABLE(R8)      ; DIRECTORIES ALWAYS CONTAIN THEMSELVES
09 A7 58 D0 OB94 1018      MOVL R8,LNMTH$L_NAME(R7)      ; STORE LNMB ADDRESS IN TABLE HEADER
19 A7 57 D0 OB98 1019      MOVL R7,LNMTH$L_QTABLE(R7)      ; DIRECTORIES ARE QUOTA HOLDERS
00000000'9F 58 D0 OB9C 1020      MOVL R8,@#CTL$GL_LNMDIRECT      ; STORE ADDR OF PROCESS DIRECTORY AWAY
OBA3 1021
51 11 A8 9E OBA3 1022      MOVAB LNMB$T_NAME(R8),R1      ; RETRIEVE THE SIZE AND ADDRESS OF THE
50 81 9A OBA7 1023      MOVZBL (R1)+,R0      ; PROCESS DIRECTORY'S NAME
00000000'GF 16 OBAA 1024      JSB G^LNMSHASH      ; HASH THE DIRECTORY NAME
OB80 1025
50 63 CA OB80 1026      BICL2 LNMHSH$L_MASK(R3),R0      ; MODIFY THE HASH INDEX TO BE IN RANGE
0C A340 58 D0 OB83 1027      MOVL R8,LNMHSH$C_BUCKET(R3)[R0] ; INSERT THE PROCESS DIRECTORY TABLE
04 A8 0C A340 DE OB88 1028      MOVAL LNMHSH$C_BUCKET(R3)[R0],- ; INTO THE APPROPRIATE HASH BUCKET
OB8E 1029
OB8E 1030
OB8E 1031
OB8E 1032
OB8E 1033
OB8E 1034
OB8E 1035
51 58 A8 9E OB8E 1036      MOVAB PROC_TABLE(R8),R1      ; COMPUTE ADDRESS OF LNM$PROCESS_TABLE
```

ALLOCATE P1 SPACE FOR THE PROCESS-PRIVATE LOGICAL NAME HASH TABLE, FOR THE PROCESS DIRECTORY LOGICAL NAME TABLE, AND FOR ALL PROCESS-PRIVATE LOGICAL NAMES AND LOGICAL NAME TABLES THAT NEED TO BE SETUP AT PROCESS CREATION TIME. INITIALLY FORMAT THE LOGICAL NAMES AND LOGICAL NAME TABLES BY COPYING THEIR TEMPLATES ONTO THE P1 SPACE ALLOCATED FOR THEM, AND THEN FORMAT THE PROCESS-PRIVATE LOGICAL NAME HASH TABLE.

FIXUP THE PROCESS DIRECTORY LOGICAL NAME TABLE, LNM\$PROCESS DIRECTORY, AND LINK IT INTO THE APPROPRIATE HASH BUCKET OF THE PROCESS-PRIVATE LOGICAL NAME HASH TABLE.

FIXUP THE PROCESS LOGICAL NAME TABLE, LNM\$PROCESS TABLE, AND INSERT IT INTO THE APPROPRIATE HASH BUCKET OF THE PROCESS-PRIVATE LOGICAL NAME HASH TABLE.

```
59 0080 C8 9E OBC2 1037 MOVAB PROC TABLE_LNMTH(R8),R9 : COMPUTE AND SAVE ADDRESS OF LNMTH
    OC A1 57 D0 OBC7 1038 MOVL R7,LNMB$TABLE(R1) : STORE CONTAINING TABLE HEADER'S ADDR
    01 A9 53 D0 OBCB 1039 MOVL R3,LNMB$HASH(R9) : STORE HASH TABLE ADDR IN TABLE HEADER
    09 A9 51 D0 OBCF 1040 MOVL R1,LNMB$NAME(R9) : STORE LNMB ADDRESS IN TABLE HEADER
    0D A9 57 D0 OBD3 1041 MOVL R7,LNMB$PARENT(R9) : LNMB$PROCESS DIRECTORY IS PARENT AND
    19 A9 57 D0 OBD7 1042 MOVL R7,LNMB$QTABLE(R9) : QUOTA HOLDER OF LNMB$PROCESS TABLE
    00000000'GF 16 D4 OBD8 1043 CLRL R2 : NO SPECIAL INSERTION ATTRIBUTES
    OBE3 1044 JSB G^LNMB$INSLOGTAB : APPROPRIATELY INSERT LNMB$PROCESS_TABLE
    OBE3 1045
    OBE3 1046 :
    OBE3 1047 : FIXUP LNMB$PROCESS LNMB$GROUP AND LNMB$JOB AND INSERT THEM INTO THE APPROPRIATE
    OBE3 1048 : HASH BUCKET OF THE PROCESS-PRIVATE LOGICAL NAME HASH TABLE. LNMB$GROUP AND
    OBE3 1049 : LNMB$JOB REQUIRE THAT THEIR EQUIVALENCE STRINGS BE CONSTRUCTED FROM THE UIC
    OBE3 1050 : AND JIB ADDRESS OF THE NEW PROCESS RESPECTIVELY.
    OBE3 1051 :
    OBE3 1052 :
    51 54 6E D0 OBE3 1053 MOVL (SP),R4 : RESTORE PCB ADDRESS TO R4
    OC 00A8 C8 9E OBE6 1054 MOVAB PROCESS(R8),R1 : COMPUTE ADDRESS OF LNMB$PROCESS
    A1 57 D0 OBE8 1055 MOVL R7,LNMB$TABLE(R1) : STORE CONTAINING TABLE HEADER'S ADDR
    00000000'GF 16 D4 OBEF 1056 CLRL R2 : NO SPECIAL INSERTION ATTRIBUTES
    OBF1 1057 JSB G^LNMB$INSLOGTAB : APPROPRIATELY INSERT LNMB$PROCESS
    OBF7 1058
    51 0118 C8 9E OBF7 1059 MOVAB JOB(R8),R1 : COMPUTE ADDRESS OF LNMB$JOB
    OC A1 57 D0 OBF8 1060 MOVL R7,LNMB$TABLE(R1) : STORE CONTAINING TABLE HEADER'S ADDR
    53 2E A1 9E OC00 1061 MOVAB JOB_XEND_SIZE-1(R1),R3 : COMPUTE ADDRESS OF LAST LNM
    52 D4 OC04 1062 CLRL R2 : CLEAR INDEX REGISTER
50 0080 C4 04 52 EF OC06 1063 60$: EXTZV R2,#4,PCB$JIB(R4),R0 : EXTRACT OUT HEX BITS AND TRANSFORM
    73 F40A CF40 90 OC0D 1064 MOVB CHARS[R0],-TR3) : THEM INTO THEIR ASCII EQUIVALENT
    FFED 52 04 1F 9D OC13 1065 ACBB #31,#4,R2,60$ : CONTINUE FROM RIGHT -> LEFT UNTIL DONE
    5A 53 D0 OC19 1066 MOVL R3,R10 : SAVE THE ADDRESS OF THE ASCII JIB ADDR
    52 D4 OC1C 1067 CLRL R2 : NO SPECIAL INSERTION ATTRIBUTES
    00000000'GF 16 OC1E 1068 JSB G^LNMB$INSLOGTAB : APPROPRIATELY INSERT LNMB$JOB
    OC24 1069
    51 00E0 C8 9E OC24 1070 MOVAB GROUP(R8),R1 : COMPUTE ADDRESS OF LNMB$GROUP
    OC A1 57 D0 OC29 1071 MOVL R7,LNMB$TABLE(R1) : STORE CONTAINING TABLE HEADER'S ADDR
    53 30 A1 9E OC2D 1072 MOVAB GROUP_XEND_SIZE-1(R1),R3 : COMPUTE ADDRESS OF LAST LNM
    52 D4 OC31 1073 CLRL R2 : CLEAR INDEX REGISTER
50 00BE C4 03 52 EF OC33 1074 61$: EXTZV R2,#3,PCB$W_GRP(R4),R0 : EXTRACT OUT OCTAL BITS AND TRANSFORM
    73 F3DD CF40 90 OC3A 1075 MOVB CHARS[R0],-TR3) : THEM INTO THEIR ASCII EQUIVALENT
    FFED 52 03 0E 9D OC40 1076 ACBB #14,#3,R2,61$ : CONTINUE FROM RIGHT -> LEFT UNTIL DONE
    73 30 90 OC46 1077 MOVB #^A/0/,-(R3) : ASSUME HIGH ORDER BIT IS 0
    03 00BE C4 0F E1 OC49 1078 BBC #15,PCB$W_GRP(R4),62$ : IF SO THEN GO INSERT LNMB$GROUP
    63 31 90 OC4F 1079 MOVB #^A/1/,-(R3) : OTHERWISE INSERT A 1
    5B 53 D0 OC52 1080 62$: MOVL R3,R11 : SAVE THE ADDRESS OF THE ASCII GROUP
    52 D4 OC55 1081 CLRL R2 : NO SPECIAL INSERTION ATTRIBUTES
    00000000'GF 16 OC57 1082 JSB G^LNMB$INSLOGTAB : APPROPRIATELY INSERT LNMB$GROUP
    OC5D 1083
    OC5D 1084 :
    OC5D 1085 : FIXUP THE LOGICAL NAME BLOCKS FOR SYSS$INPUT, TT, SYSS$OUTPUT, SYSS$ERROR, AND
    OC5D 1086 : SYSS$DISK, AND INSERT THEM INTO THE APPROPRIATE HASH BUCKET OF THE
    OC5D 1087 : PROCESS-PRIVATE LOGICAL NAME HASH TABLE.
    OC5D 1088 :
    OC5D 1089 :
    OC5D 1090 CRELNM - : FIXUP AND INSERT SYSS$INPUT
    OC5D 1091 PQBST_INPUT,-
    OC5D 1092 PQBSL_INPUT_ATT,-
    OC5D 1093 SYSS$INPUT_LNM,-
```

```
0C5D 1094 SYSS$INPUT
0C68 1095
0C68 1096 CRELM - ; FIXUP AND INSERT SYSS$OUTPUT
0C68 1097 PQBST_OUTPUT,-
0C68 1098 PQBSL_OUTPUT_ATT,-
0C68 1099 SYSS$OUTPUT_LNM,-
0C68 1100 SYSS$OUTPUT
0C73 1101
0C73 1102 CRELM - ; FIXUP AND INSERT SYSS$ERROR
0C73 1103 PQBST_ERROR,-
0C73 1104 PQBSL_ERROR_ATT,-
0C73 1105 SYSS$ERROR_LNM,-
0C73 1106 SYSS$ERROR
0C7E 1107
0C7E 1108 CRELM - ; FIXUP AND INSERT TT
0C7E 1109 PQBST_INPUT,-
0C7E 1110 PQBSL_INPUT_ATT,-
0C7E 1111 TT_LNM,-
0C7E 1112 TT
0C89 1113
0C89 1114 CRELM - ; FIXUP AND INSERT SYSS$DISK
0C89 1115 PQBST_DISK,-
0C89 1116 PQBSL_DISK_ATT,-
0C89 1117 SYSS$DISK_LNM,-
0C89 1118 SYSS$DISK
0C94 1119
0C94 1120
0C94 1121 : IF THE PROCESS BEING CREATED IS NOT A SUB-PROCESS THEN CREATE THE JOB AND
0C94 1122 : GROUP LOGICAL NAME TABLES.
0C94 1123 :
0C94 1124
54 8E D0 0C94 1125 MOVL (SP)+,R4 ; RETRIEVE PCB ADDRESS
1C A4 D5 0C97 1126 TSTL PCB$OWNER(R4) ; SUB-PROCESS?
OD 12 0C9A 1127 BNEQ 65$ ; IF YES THEN SKIP TABLE CREATION
57 40 A6 D0 0C9C 1128 MOVL PQBSL_JTQUOTA(R6),R7 ; RETRIEVE JOB TABLE CREATION QUOTA
04EA 30 0CA0 1129 BSBW EXESCRE_JGTABLE ; CREATE JOB AND GROUP TABLES
03 50 E8 0CA3 1130 BLBS RO,65$ ; CONTINUE IF SUCCESS
FE2C 31 0CA6 1131 64$: BRW VABUG ; OTHERWISE, TAKE COMMON EXIT PATH
0CA9 1132
0CA9 1133 :
0CA9 1134 : ALLOCATE P1 SPACE FOR THE PROCESS-PRIVATE LOGICAL NAME TABLE NAME CACHE
0CA9 1135 :
0CA9 1136
51 00000000'GF 08 C5 0CA9 1137 65$: MULL3 #8,G^LNMSGL_HTBLSIZP,R1 ; ALLOCATE TWICE HASH TABLE SIZE
58 51 00000080 8F C7 0CB1 1138 DIVL3 #LNMC$K_LENGTH,R1,R8 ; COMPUTE # OF ENTRIES
23 13 0CB9 1139 BEQL 67$ ; IF ANY
00000000'GF 16 0CBB 1140 JSB G^EXESALOP1PROC ; ALLOCATE TOTAL AMOUNT OF SPACE NEEDED
E2 50 E9 OCC1 1141 BLBC RO,64$ ; IF POSSIBLE
08 A2 0080 8F B0 OCC4 1142 66$: MOVW #LNMC$K_LENGTH,LNMC$W_SIZE(R2) ; SET SIZE
OC A2 D4 OCCA 1143 CLRL LNMC$L_TBLADDR(R2) ; MARK EMPTY
00000000'9F 62 OE OCCD 1144 INSQUE (R2),@CTL$GQ_LNMTBLCACHE ; INSERT IN QUEUE
52 00000080 8F C0 OCD4 1145 ADDL2 #LNMC$K_LENGTH,R2 ; POINT TO NEXT
E6 58 F5 OCDB 1146 SOBGTR R8,66$ ; LOOP
OCDE 1147
OCDE 1148 :
OCDE 1149 : RESTORE PCB AND PHD ADDRESS, SET IPL TO 0 TO ALLOW FOR PROCESS DELETION
OCDE 1150 : (IF DESIRED), RESET ADDRESS SPACE, AND SET WSLAST.
```

```

      55 8E D0 OCDE 1151 ;
      5C 00000000'9F DE OCE1 1152
      00000000'GF 16 OCE1 1153 67$: MOVL (SP)+,R5 ; RESTORE PHD ADDRESS
      OCE8 1154
      OCEE 1155 MOVAL @#MMG$IMGHDRBUF,AP ; IMAGE HEADER BUFFER ADDRESS
      OCEE 1156 JSB G*MMG$IMGRESET ; RESET ADDRESS SPACE AND SET WSLAST
      OCEE 1157
      OCEE 1158 ; THE FOLLOWING MOVC SEQUENCES DESTROY R0 THROUGH R5
      OCEE 1159
      6C 07C8 C6 9A OCEE 1160 IMGNAM: MOVZBL PQBST_IMAGE(R6), (AP) ; SIZE OF IMAGE NAME STRING
      04 AC 08 AC DE OCF3 1161 MOVAL 8(AP), 4(AP) ; ADDRESS OF IMAGE NAME STRING
      08 AC 07C9 C6 6C 28 OCF8 1162 MOVC3 (AP), PQBST_IMAGE+1(R6), 8(AP) ; MOVE THE NAME STRING
      OCF8 1163
      06C8 C6 95 OCF8 1164 TSTB PQBST_DDSTRING(R6) ; CHECK FOR NULL STRING
      0C 13 OD03 1165 BEQL 70$ ; YES, DONT MOVE ANYTHING
      00000000'9F 0100 8F 28 OD05 1166 MOVC3 #PQB$$_DDSTRING, -
      06C8 C6 OD09 1167 PQBST_DDSTRING(R6), @#PIO$GT_DDSTRING ; AND DEFAULT DIRECTORY
      OD11 1168 70$: ; CONTINUE
      OD11 1169
      OD11 1170 ; Move CLI and CLI table information to P1 space in one fell swoop:
      OD11 1171 PQBST_CLI_NAME -> CTL$GT_CLINAME
      OD11 1172 PQBST_CLI_TABLE -> CTL$GT_TABLENAME
      OD11 1173 PQBST_SPAWN_CLI -> CTL$GT_SPAWNCLI
      OD11 1174 PQBST_SPAWN_TABLE -> CTL$GT_SPAWNTABLE
      OD11 1175
      OD11 1176 ASSUME PQBST_CLI_TABLE EQ <PQBST_CLI_NAME + PQB$$_CLI_NAME>
      OD11 1177 ASSUME PQBST_SPAWN_CLI EQ <PQBST_CLI_TABLE + PQB$$_CLI_TABLE>
      OD11 1178 ASSUME PQBST_SPAWN_TABLE EQ <PQBST_SPAWN_CLI + PQB$$_SPAWN_CLI>
      OD11 1179
      28 OD11 1180 MOVC3 #<PQB$$_CLI_NAME+-
      OD12 1181 PQB$$_CLI_TABLE+-
      OD12 1182 PQB$$_SPAWN_CLI+-
      OD12 1183 PQB$$_SPAWN_TABLE>, -
      OD12 1184 PQBST_CLI_NAME(R6), @#CTL$GT_CLINAME
      OD18
      OD1D 1185
      OD1D 1186 ; STORE EVERYTHING ELSE OF INTEREST BEFORE WE GET RID OF THE PQB
      OD1D 1187
      00000000'9F 4C A6 D0 OD1D 1188 MOVL PQB$$_CREPRC_FLAGS(R6), @#CTL$GL_CREPRC_FLAGS
      00000000'9F 48 A6 D0 OD25 1189 MOVL PQB$$_UAF_FLAGS(R6), @#CTL$GL_UAF_FLAGS
      OD2D 1190
      OD2D 1191 ***** TEMP *****
      OD2D 1192
      OD2D 1193 THE FOLLOWING CODE WILL BE REMOVED WHEN WE DECIDE WHAT TO DO WITH THE
      OD2D 1194 ACCOUNT AND USERNAME FIELDS IN THE P1 POINTER PAGE.
      OD2D 1195
      OD2D 1196 assume jib$$_account eq <jib$$_username + jib$$_username>
      OD2D 1197
      50 00000000'GF D0 OD2D 1198 movl g*ctl$gl_pcb, r0 ; get pcb address ...
      50 0080 C0 D0 OD34 1199 movl pcb$$_jib(r0), r0 ; so that we can get jib address
      14 28 OD39 1200 movc3 #<jib$$_username + jib$$_account>, -
      0C A0 OD3B 1201 jib$$_username(r0), - ; move username and account
      00000000'9F OD3D 1202 @#ctl$$_username ; in one instruction
      OD42 1203
      OD42 1204 ***** END TEMP *****
      OD42 1205
      00000000'FF 66 OE OD42 1206 INSQUE (R6), @EXE$GL_PQBBL ; DEALLOCATE PQB TO LOOKASIDE LIST
```

```
00000000'9F 00000000'9F DE 0D49 1207 SETIPL #0 ; DROP IPL AND ALLOW PROCESS DELETION
00000000'9F 00000000'9F DE 0D4C 1208
0D4C 1209
0D4C 1210 : INITIALIZE FIXUP VECTOR LINKED LISTS TO CONTAIN A SINGLE DUMMY ENTRY
0D4C 1211 :
0D4C 1212
0D4C 1213 MOVAL @#CTL$GL_IAPERM,@#CTL$GL_I AFLINK
0D57 1214 MOVAL @#CTL$GL_IAPERM,@#CTL$GL_I AFLAST
0D62 1215
0D62 1216 :
0D62 1217 : INITIALIZE ARRAYS THAT DETERMINE HOW PRIVILEGED VECTORS ARE RESET
0D62 1218 :
50 00000000'9F 3E 0D62 1219 MOVAV @#IAC$AW_VECRESET,R0 ; STORE RESET ARRAY ADDRESS
80 04 B0 0D69 1220 MOVW #4,(R0)+ ; KERNEL VECTOR
80 04 B0 0D6C 1221 MOVW #4,(R0)+ ; EXEC VECTOR
80 04 B0 0D6F 1222 MOVW #4,(R0)+ ; RUNDOWN VECTOR
80 04 B0 0D72 1223 MOVW #4,(R0)+ ; MESSAGE VECTOR
0D75 1224
50 00000000'9F 3E 0D75 1225 MOVAV @#IAC$AW_VECSET,R0 ; STORE START ARRAY ADDRESS
80 04 B0 0D7C 1226 MOVW #4,(R0)+ ; KERNEL VECTOR
80 04 B0 0D7F 1227 MOVW #4,(R0)+ ; EXEC VECTOR
80 04 B0 0D82 1228 MOVW #4,(R0)+ ; RUNDOWN VECTOR
80 04 B0 0D85 1229 MOVW #4,(R0)+ ; MESSAGE VECTOR
0D88 1230
0D88 1231 EXESPROCIMACT: ; ENTRY POINT FOR STAND-ALONE SYSGEN
58 54 00000000'9F D0 0D88 1232 MOVL @#CTL$GL_PCB,R4 ; GET PCB ADDRESS
24 A4 01 13 EF 0D8F 1233 EXTZV #PCBSV_HIBER,#1,PCBSL_STS(R4),R8; SAVE HIBERNATE CONTROL
00000F4C'EF 00 FB 0D95 1234 CALLS #0,XQPMERGE ; MERGE XQP INTO PROCESS
03 00000000'GF 00' E1 0D9C 1235 BBC S^#EXESV_INIT,G^EXESGL_FLAGS,72$ ; DON'T MERGE IF NOT INIT
5A 50 E9 0DA4 1236 BLBC R0,75$ ; EXIT IF MERGE FAILS
7E 05 16 9C 0DA7 1237 72$: ROTL #PSL$V_PVMOD,#<PSL$C_EXEC@2+PSL$C_EXEC>,-(SP) ; FORM EXEC PSL
6C 10 0DAB 1238 BSBB 80$ ; CHANGE MODE TO EXECUTIVE
0DAD 1239
0DAD 1240 ; ***** THE FOLLOWING CODE EXECUTES IN EXEC MODE *****
0DAD 1241
52 6C 9A 0DAD 1242 MOVZBL (AP),R2 ; GET ADR OF FILENAME STRING DESC
52 03 C0 0DB0 1243 ADDL #3,R2 ; ROUND THE NUMBER OF BYTES IN
52 03 CA 0DB3 1244 BICL #3,R2 ; THE NAME UP TO A LONGWORD BOUNDARY
5E 52 C2 0DB6 1245 SUBL R2,SP ; ALLOCATE SPACE FOR NAME ON STACK
6E 9F 0DB9 1246 PUSHAB (SP) ; BUILD STRING DESCRIPTOR FOR
7E 6C 9A 0DBB 1247 MOVZBL (AP),-(SP) ; FILENAME ON THE STACK
51 5E D0 0DBE 1248 MOVL SP,R1 ; GET ADR OF STRING DESCRIPTOR
3E BB 0DC1 1249 PUSHR #^M<R1,R2,R3,R4,R5> ; SAVE REGISTERS
04 B1 04 BC 52 28 0DC3 1250 MOV C3 R2,@4(AP),@4(R1) ; MOVE FILENAME TO STACK
3E BA 0DC9 1251 POPR #^M<R1,R2,R3,R4,R5> ; RESTORE REGISTERS
0DCB 1252 $IMGACT,S - ; ACTIVATE THE IMAGE
0DCB 1253 -NAME =(AP),- ; DESCRIPTOR FOR IMAGE NAME
0DCB 1254 DFLNAM=DEFDESC,- ; DEFAULT NAME DESCRIPTOR
0DCB 1255 HDRBUF=(AP) ; ADDRESS IF IMAGE HEADER BUFFER
52 08 C0 0DE2 1256 ADDL #8,R2 ; CALCULATE # OF BYTES ON STACK
5E 52 C0 0DE5 1257 ADDL R2,SP ; AND CLEAN THEM OFF
16 50 E9 0DE8 1258 BLBC R0,75$ ; BRANCH IF IMACT FAILED
50 00000000'9F 9E 0DEB 1259 MOVAB @#PIOSAL_RMSEXH,R0 ; GET ADDRESS OF EXIT HANDLER CONTROL BLOCK
04 A0 0F25'CF 9E 0DF2 1260 MOVAB W^EXESRMSEXH,4(R0) ; SET ADDRESS OF RMS EXIT HANDLER
0DF8 1261 $DCLEXH,S (R0) ; DECLARE EXEC MODE EXIT HANDLER
00000000'GF 63 50 E9 0E01 1262 75$: BLBC -R0,120$ ; IF LBC ERROR
0E6B'CF 9E 0E04 1263 MOVAB W^EXESCLI_UTILSRV+2,G^CTL$AL_CLICALBK ; SET CLI CALL BACK ADDRESS
```

```
7E 0F 16 9C 0E0D 1264      ROTL    #PSL$V_PVMOD,#<PSL$C_USER@2+PSL$C_USER>,-(SP) ; FORM USER PSL
      06 10 0E11 1265      BSBB     B0$ ; CHANGE TO USER MODE
      0E13 1266
      0E13 1267 ; ***** THE FOLLOWING CODE EXECUTES IN USER MODE *****
      0E13 1268
      1A'AF 5D D4 0E13 1269      CLRL    FP ; TERMINATE CALL FRAME CHAIN
      6C FA 0E15 1270      CALLG   (AP),B*90$ ; CREATE TOP FRAME
      02 0E19 1271 80$:      REI      ; CHANGE TO NEW MODE
      0000 0E1A 1272 90$:      .WORD   0 ; ENTRY MASK
      6D 80'AF 9E 0E1C 1273      MOVAB   B*EXESCATCH_ALL,(FP) ; SET EXCEPTION HANDLER ADDRESS
      0E20 1274      $SETEXV_S #2,B*EXESCATCH_ALL ; DECLARE LAST CHANCE HANDLER
      0E30 1275      $IMGFIX-S ; PERFORM ADDRESS RELOCATION
      2D 50 E9 0E37 1276      BLBC     R0,120$ ; QUIT IF ERROR OCCURS
      58 DD 0E3A 1277      PUSHL    R8 ; SAVE HIBERNATE FLAG
      52 6C 7D 0E3C 1278 100$:      MOVQ   (AP),R2 ; GET IMAGE HEADER BLOCK DESCRIPTOR
      7E D4 0E3F 1279      CLRL     -(SP) ; CLEAR COMMAND INTERPRETER FLAGS
      20 A2 DD 0E41 1280      PUSHL    IHD$L_LNKFLAGS(R2) ; PUSH LINKER FLAGS
      7E 52 7D 0E44 1281      MOVQ   R2,-(SP) ; THIRD AND FOURTH ARGUMENTS TO PROG
      69'AF 9F 0E47 1282      PUSHAB  B*EXESCLI_UTILSRV ; PUSH ADDRESS OF CLI CALL BACK ROUTINE
      51 02 A2 3C 0E4A 1283      MOVZWL  IHD$W_ACTIVOFF(R2),R1 ; OFFSET TO TRANSFER VECTOR
      52 51 C0 0E4E 1284      ADDL     R1,R2 ; FORM ADDRESS OF START VECTOR
      62 DF 0E51 1285      PUSHAL   (R2) ; MOVE TO ARGUMENT LIST
      07 18 AE E9 0E53 1286      BLBC     24(SP),110$ ; BR IF NO HIBERNATE
      92 06 FB 0E5E 1287 110$:      $HIBER-S ; SET, HIBERNATE UNTIL SOME WAKE
      03 50 E9 0E61 1289      CALLS   #6,@(R2)+ ; CALL IMAGE
      D5 6E E8 0E64 1290      BLBC     R0,120$ ; EXIT IF NOT SUCCESS
      0A 11 0E67 1291 120$:      BRB     EXES$EXIT_IMAGE ; CHECK FOR HIBERNATE AGAIN
      0E69 1292
      0E69 1293 ;
      0E69 1294 ; DUMMY COMMAND INTERPRETER CALL BACK ROUTINE
      0E69 1295 ;
      0E69 1296
      50 00038822 8F 0000 0E69 1297      .ENTRY  EXESCLI_UTILSRV,*M<>
      D0 0E6B 1298      MOVL     #CLIS_INVREQTYP,R0 ; SET INVALID REQUEST TYPE STATUS
      04 0E72 1299      RET      ;
      0E73 1300      .DSABL  LSB ;
```

```

OE73 1303 .SBTTL EXIT IMAGE AND RUN DOWN FILES
OE73 1304 :+
OE73 1305 :
OE73 1306 EXE$EXIT_IMAGE - EXIT IMAGE AND RUN DOWN FILES
OE73 1307 :
OE73 1308 THIS ROUTINE IS JUMPED TO AT THE CONCLUSION OF IMAGE EXECUTION TO RUN DOWN
OE73 1309 RMS FILES AND TO RETURN THE FINAL IMAGE STATUS.
OE73 1310 :
OE73 1311 INPUTS:
OE73 1312 :
OE73 1313 RO = FINAL IMAGE STATUS.
OE73 1314 :
OE73 1315 OUTPUTS:
OE73 1316 :
OE73 1317 IMAGE EXIT IS EXECUTED.
OE73 1318 :-
OE73 1319 :
OE73 1320 EXE$EXIT_IMAGE::
OE73 1321 PUSH  R0
OE73 1322 PUSH  #1
00000000'9F 50 DD OE75 1323 10$: CALL  (SP),@#SYS$EXIT
6E FA OE77 1324 BRB   10$
F7 11 OE7E 1324

```

```

: EXIT IMAGE
: SAVE FINAL IMAGE STATUS
: SET NUMBER OF ARGUMENTS
: EXIT IMAGE
:

```

```
0E80 1327 .SBTTL CATCH ALL CONDITION HANDLER
0E80 1328
0E80 1329 *
0E80 1330 EXESATCH_ALL - CATCH ALL CONDITION HANDLER
0E80 1331 THIS ROUTINE IS ENTERED AS THE RESULT OF AN UNFIELDER OR IMPROPERLY HANDLED
0E80 1332 EXCEPTION CONDITION OR SOFTWARE SIGNAL.
0E80 1333
0E80 1334 INPUTS:
0E80 1335
0E80 1336 CHFSL_MCHARGLST(AP) = ADDRESS OF MECHANISM ARGUMENT LIST.
0E80 1337 CHFSL_SIGARGLST(AP) = ADDRESS OF CONDITION ARGUMENT LIST.
0E80 1338
0E80 1339 OUTPUTS:
0E80 1340
0E80 1341 A MESSAGE IS ISSUED USING THE SYSS$PUTMSG SYSTEM SERVICE AND A TEST IS
0E80 1342 MADE ON THE CONDITION NAME TO DETERMINE IF THE IMAGE SHOULD BE ALLOWED
0E80 1343 TO CONTINUE EXECUTION. THE FOLLOWING CONDITIONS CAUSE A FORCED IMAGE
0E80 1344 EXIT:
0E80 1345
0E80 1346 1. ANY ENTRY TO THIS ROUTINE VIA THE LAST CHANCE VECTOR.
0E80 1347
0E80 1348 2. THE CONDITION NAME HAS A SEVERITY OF 'SEVERE ERROR'.
0E80 1349
0E80 1350 IF A FORCED IMAGE EXIT IS PERFORMED, THEN A SUMMARY OF THE CONDITION
0E80 1351 ARGUMENTS AND FINAL REGISTERS ARE WRITTEN TO SYSS$OUTPUT.
0E80 1352
0E80 1353
0E80 1354 .ENTRY EXESATCH_ALL,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
0E82 1355 PUSHL #0 ; SET EXCEPTION NAME FLAG FALSE
0E84 1356 PUSHL R2 ; SAVE REGISTER
0E86 1357 MOVL CHFSL_SIGARGLST(AP),R2 ; GET ADDRESS OF SIGNAL ARGUMENTS
0E8A 1358 PUSHL (R2) ; SAVE NUMBER OF ARGUMENTS
0E8C 1359 CMPW CHFSL_SIG_NAME(R2),#SS$SSFAL ; IS EXCEPTION SYS. SERV. FAIL.?
0E92 1360 BNEQ SS$ ; NO
0E94 1361 $SETSM_S #0 ; YES, TURN OFF SYS. SERV. FAIL. EXCEP.
0E9D 1362 5$: TSTW CHFSL_SIG_NAME+2(R2) ; POSSIBLY SYSTEM EXCEPTION NAME?
0EA0 1363 BNEQ 20$ ; IF NEQ NO
0EA2 1364 INCL 8(SP) ; SET EXCEPTION NAME FLAG TRUE
0EA5 1365 MOVAB L^EXE$EXCEPTABLE,R1 ; GET ADDRESS OF EXCEPTION TABLE
0EAC 1366 MOVZBL (R1)+,R0 ; SET LOOP COUNT
0EAF 1367 10$: TSTB (R1)+ ; SKIP NUMBER OF ARGUMENTS
0EB1 1368 MOVZWL (R1)+,-(SP) ; GET NEXT HARDWARE EXCEPTION CODE
0EB4 1369 CMPZV #STSSV_CODE,#STSS$CODE ; CONDITION VALUE HARDWARE CODE?
0EB7 1370 CHFSL_SIG_NAME(R2),-(SP)+ ;
0EBA 1371 BEQL 30$ ; IF EQL YES
0EBC 1372 SOBGTR R0,10$ ; ANY MORE TO COMPARE?
0EBF 1373 CLRL 8(SP) ; SET EXCEPTION NAME FLAG FALSE
0EC2 1374 20$: SUBL #2,(R2) ; ADJUST LENGTH OF ARGUMENT LIST
0EC5 1375 30$: TSTB @#CTL$GB_SSFILTER ; SYSTEM SERVICE INHIBITED NOW?
0ECB 1376 BNEQ 35$ ; YES, DO NOT TRY TO PRINT ANYTHING
0ECD 1377 PUSHL #0 ; CLEAR ADDRESS OF FACILITY NAME DESCRIPTOR
0ECF 1378 PUSHL #0 ; CLEAR ADDRESS OF ACTION ROUTINE
0ED1 1379 PUSHAB (R2) ; SET ADDRESS OF MESSAGE VECTOR
0ED3 1380 CALLS #3,@#SYSS$PUTMSG ; OUTPUT MESSAGE
0EDA 1381 35$: POPL (R2) ; RESTORE ARGUMENT COUNT
0EDD 1382 MOVL CHFSL_SIG_NAME(R2),R0 ; GET CONDITION NAME
0EE1 1383 POPL R2 ; RESTORE REGISTER
```

52	04	AC	DD	0E82	1355
045C	8F	04	A2	0E84	1356
		09	B1	0E86	1357
			12	0E8A	1358
				0E8C	1359
				0E92	1360
				0E94	1361
	06	A2	B5	0E9D	1362
		20	12	0EA0	1363
	08	AE	D6	0EA2	1364
51	00000000	'EF	9E	0EA5	1365
	50	81	9A	0EAC	1366
		81	95	0EAF	1367
	7E	81	3C	0EB1	1368
	0C	03	ED	0EB4	1369
	8E	04	A2	0EB7	1370
		09	13	0EBA	1371
	F0	50	F5	0EBC	1372
	08	AE	D4	0EBF	1373
	62	02	C2	0EC2	1374
	00000000	'9F	95	0EC5	1375
		0D	12	0ECB	1376
		00	DD	0ECD	1377
		00	DD	0ECF	1378
		62	9F	0ED1	1379
	00000000	'9F	03	0ED3	1380
		62	8ED0	0EDA	1381
	50	04	A2	0EDD	1382
		52	8ED0	0EE1	1383

```
7E 51 08 AC D0 0EE4 1384      MOVL    CHFSL_MCHARGLST(AP),R1      : GET ADDRESS OF MECHANISM ARRAY
    08 A1 03 C1 0EE8 1385      ADDL3   #3,CHFSL_MCH_DEPTH(R1),-(SP) : LAST CHANCE ENTRY?
    07 0E 13 0EED 1386      BEQL     50$                          : IF EQL YES
    03 00 E8 0EEF 1387      BLBS     R0,40$                      : IF LBS SUCCESS CODE
    04 50 ED 0EF2 1388      CMPZV    #STSSV_SEVERITY,#STSS$_SEVERITY,- : SEVERE ERROR OR GREATER?
    04 50 18 0EF3 1389      R0,#STSS$_SEVERE                      :
    50 01 3C 0EF7 1390      BGEQ     50$                          : IF GEQ YES
    04 04 3C 0EF9 1391 40$:  MOVZWL   #SS$_CONTINUE,R0           : SET CONTINUATION CODE
    04 04 04 0EFC 1392      RET                                     :
    50 DD 0EFD 1393      :
    00000000'9F 95 0EFF 1394 50$:  PUSHL   R0                      : SAVE EXCEPTION NAME
    14 12 0F05 1395      TSTB     @#CTL$GB_SSFILTER              : SYSTEM SERVICES INHIBITED NOW?
    0D 08 AE E9 0F07 1396      BNEQ   70$                          : YES, DON'T TRY TO PRINT ANYTHING
    6C 9F 0F0B 1397      BLBC     8(SP),60$                      : IF LBC NOT EXCEPTION
    F11B CF 9F 0F0D 1398      PUSHAB  (AP)                        : SET ADDRESS OF SIGNAL ARGUMENTS
    00000000'EF 02 FB 0F11 1400     PUSHAB  SUFFIX                 : SET ADDRESS OF MESSAGE SUFFIX
    00D2 30 0F18 1401 60$:  CALLS    #2,EXESEXCMMSG               : OUTPUT EXCEPTION SUMMARY
    50 8ED0 0F1B 1402 70$:  BSBW     EXE$IMGDMP_MERGE             : TRY TO TAKE A DUMP
    00 50 1C E2 0F1E 1403     POPL    R0                          : RESTORE EXCEPTION NAME
    FF4E 31 0F22 1404 80$:  BBSS     #STSSV_INHIB_MSG,R0,80$       : SET INHIBIT MESSAGE BIT
    BRW     EXE$EXIT_IMAGE
```

```
OF25 1407 .SBTTL EXESRMSEXH - EXEC Mode Exit Handler
OF25 1408
OF25 1409 :+ EXESRMSEXH - Executive mode exit handler
OF25 1410
OF25 1411 : This routine is called as the result of an attempt to exit from exec mode.
OF25 1412 : It's function is to run down all RMS files.
OF25 1413
OF25 1414 : INPUTS:
OF25 1415
OF25 1416 : NONE.
OF25 1417
OF25 1418 : OUTPUTS:
OF25 1419
OF25 1420 : NONE.
OF25 1421
OF25 1422 : SIDE EFFECTS:
OF25 1423
OF25 1424 : RMS files are run down.
OF25 1425 :-
OF25 1426
OF25 1427 .ENTRY EXESRMSEXH,^M<>
OF25 1428 MOVAB -128(SP),SP
OF25 1429 PUSHAB (SP)
OF25 1430 PUSHL #0
OF25 1431 10$: MOVZBL #128,(SP)
OF25 1432 PUSHL #1
OF25 1433 PUSHAB 4(SP)
OF25 1434 CALLS #2,@#SYSS$RMSRUNDN
OF25 1435 CMPL R0,#RMS$_BUSY
OF25 1436 BEQL 20$
OF25 1437 BLBC R0,10$
OF25 1438 20$: RET
OF25 1439
```

5E 80 AE 0000 9E OF27 1428  
6E 80 8F 00 DD OF28 1429  
04 AE 01 DD OF2F 1431  
00000000'9F 02 FB OF35 1433  
0001848C 8F 50 D1 OF38 1434  
03 13 OF3F 1435  
E4 50 E9 OF46 1436  
04 OF48 1437  
OF4B 1438  
OF4C 1439

: ALLOCATE STRING BUFFER  
: BUILD BUFFER DESCRIPTOR  
: SET LENGTH OF STRING BUFFER  
: RUN DOWN IMAGE AND ALL PPFS  
: PUSH ADDRESS OF BUFFER DESCRIPTOR  
: RUN DOWN THE NEXT FILE  
: BUSY ERROR IMPLIES DON'T TRY  
: TO DO RUNDOWN AT ALL  
: IF LBC, THEN MORE TO GO

```
OF4C 1442 .SBTTL XQPMERGE - Merge the XQP into P1 Space
OF4C 1443 **
OF4C 1444 FUNCTIONAL DESCRIPTION:
OF4C 1445
OF4C 1446 This routine merges the XQP into P1 space.
OF4C 1447
OF4C 1448 The number of global sections specified by XQP$GL_SECTIONS is mapped into
OF4C 1449 the end of P1 space. The sections have names of the form SYSXQP_nnn where
OF4C 1450 nnn ranges from zero to XQP$W_SECTIONS-1. The section is mapped writeable-CRF
OF4C 1451 if the corresponding bit in XQP$GL_SECPROT is set.
OF4C 1452
OF4C 1453 CALLING SEQUENCE:
OF4C 1454
OF4C 1455 CALLS #0,XQPMERGE
OF4C 1456
OF4C 1457 INPUT PARAMETERS:
OF4C 1458
OF4C 1459 NONE
OF4C 1460
OF4C 1461 IMPLICIT INPUT:
OF4C 1462
OF4C 1463 none
OF4C 1464
OF4C 1465 OUTPUT PARAMETERS:
OF4C 1466
OF4C 1467 none
OF4C 1468
OF4C 1469 IMPLICIT OUTPUT:
OF4C 1470
OF4C 1471 NONE
OF4C 1472
OF4C 1473 COMPLETION CODES:
OF4C 1474
OF4C 1475 R0 low bit set => XQP successfully merged
OF4C 1476
OF4C 1477 SS$_NORMAL
OF4C 1478
OF4C 1479 R0 low bit clear => Error occurred while merging XQP
OF4C 1480
OF4C 1481 Various errors returned by $IMGACT and $MGBLSC
OF4C 1482
OF4C 1483 SIDE EFFECTS:
OF4C 1484
OF4C 1485 The permanent portion of P1 space is
OF4C 1486 expanded to accommodate the merged image.
OF4C 1487
OF4C 1488 :--
OF4C 1489
OF4C 1490 XQPMERGE:
OF4C 1491 .WORD ^M<R2,R3,R4,R5,R6,R7> ;REGISTER SAVE MASK
OF4C 1492 :
OF4C 1493 TSTL XQP$GL_DZRO ;IS THERE ANY DZRO
OF4C 1494 BEQL 5$ ;NO
OF4C 1495 $EXPREG_5 - ;CREATE THE XQP OWN STORAGE
OF4C 1496 -PAGCNT = XQP$GL_DZRO,-
OF4C 1497 REGION = #1,-
OF4C 1498 ACMODE = #PSL$C_EXEC
```

00000000'EF D5  
16 13

```

      70 50      E9 0F69 1499      BLBC      R0,20$
      SE 0000000C'8F C2 0F6C 1500 :
      56 5E D0 0F73 1502 5$:      SUBL      #<XQP_NAMSIZ+3>8^C3,SP :RESERVE SPACE FOR GSD NAME
66 00000FDD'EF 000A'8F 28 0F76 1503      MOVL      SP,R6 :SAVE ADDRESS OF GSD NAME
      53 00000000'EF D0 0F80 1504      MOVC3     #XQP_NAMSIZ,XQP_NAM,(R6) :PUT GSD NAME IN WRITEABLE STORAGE
      00000009'E6 53 80 0F87 1505      MOVL      XQP$GL_SECTIONS,R3 :COUNT OF SECTIONS TO MAP
      56 DD 0F8E 1506      ADDB      R3,XQP_NAMSIZ-1(R6) :START WITH LAST GSD NAME
      0000000A'8F DD 0F90 1507      PUSHL     R6 :BUILD DESCRIPTOR FOR GSD NAME
      52 5E D0 0F96 1508      PUSHL     #XQP_NAMSIZ
      7E 7FFFFFFF'8F D0 0F99 1509      MOVL      SP,R2 :ADDRESS OF DESCRIPTOR
      7E 6E D0 0FA0 1510      MOVL      #^X7FFFFFFF,-(SP) :END VA FOR BLUEPRINT PO VA RANGE
      54 5E D0 0FA3 1511      MOVL      (SP),-(SP) :START VA FOR BLUEPRINT PO VA RANGE
      7E 7C 0FA6 1512      MOVL      SP,R4 :ADR OF INPUT VA RANGE
      55 5E D0 0FA8 1513      CLRQ      -(SP) :RETURN VA RANGE
      53 D7 0FAB 1514      MOVL      SP,R5 :ADR OF RETURN VA RANGE
      00000009'E6 97 0FAD 1515 10$:      DECL     R3 :MAKE COUNT ZERO-BASED
      0FB3 1516      DECB      XQP_NAMSIZ-1(R6) :NEXT GSD NAME
      0FB3 1517      $MGBLSC S -
      0FB3 1518      INADR = (R4),-
      0FB3 1519      RETADR = (R5),-
      0FB3 1520      FLAGS = #<SEC$M_EXPREG!SEC$M_SYSGBL>,-
      0FB3 1521      GSDNAM = (R2),-
      0FB3 1522      ACMODE = #PSL$C_EXEC
      OD 50      E9 0FCC 1522      BLBC      R0,20$
      DB 53      F4 0FCF 1523      SOBGEQ   R3,10$
      00000000'GF 65 D0 0FD2 1524 :
      00 B5 17 0FD9 1525      MOVL      (R5),G^CTL$GL_CTLBASVA :SET A NEW CONTROL REGION BASE
      04 0FDC 1526      JMP      @ (R5) :XQP SELF-INITIALIZATION
      0FDD 1527 20$:      RET :AND RETURN TO CALLER
      0FDD 1528
      0FDD 1529 XQP_NAM:
30 30 30 5F 50 51 58 53 59 53 0FDD 1530      .ASCII /SYSXQP 000/
      0000000A 0FE7 1531      XQP_NAMSIZ = .-XQP_NAM
```

```

OFE7 1534 .SBTTL IMAGE DUMP MERGE
OFE7 1535
OFE7 1536 *+
OFE7 1537 EXES$IMGDMP_MERGE - MERGE IN THE IMAGE DUMP FACILITY AND CALL IT
OFE7 1538
OFE7 1539 THIS ROUTINE IS ENTERED AS THE RESULT OF A FATAL CONDITION WHICH WILL FORCE
OFE7 1540 IMAGE EXIT
OFE7 1541
OFE7 1542 INPUTS:
OFE7 1543
OFE7 1544 CHF$M_MCHARGLIST(AP) = ADDRESS OF MECHANISM ARGUMENT LIST.
OFE7 1545 CHF$M_SIGARGLIST(AP) = ADDRESS OF CONDITION ARGUMENT LIST.
OFE7 1546
OFE7 1547 OUTPUTS:
OFE7 1548
OFE7 1549 AFTER PRIVILEGE CHECKS, THE IMAGE DUMP FACILITY IS MERGED INTO THE
OFE7 1550 ADDRESS SPACE AND CALLED.
OFE7 1551 :-
OFE7 1552
OFE7 1553 EXES$IMGDMP EXEC::
OFE7 1554 PUSHR #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; EXEC MODE ENTRY POINT
OFE7 1555 BRB EXEC_M
OFE7 1556
OFE7 1557 .ENABL LSB
OFE7 1558
OFE7 1559 EXES$IMGDMP MERGE::
OFE7 1560 PUSHR #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
OFE7 1561 MOVPSL R0 ; GET CURRENT PSL
OFE7 1562 EXTIV #PSL$V_CURMOD,#PSL$S_CURMOD,R0,R0
OFE7 1563 CMPL R0,#PSL$C_USER ; IS IT USER MODE
OFE7 1564 BNEQ 5$ ; NO - DUMP NOT ALLOWED
OFE7 1565 EXEC_M: MOVAL -SCRATCHSIZE(SP),SP ; RESERVE SCRATCH SPACE ON STACK
OFE7 1566 MOVL SP,R6
OFE7 1567 MOVCS #0,(SP),#0,#SCRATCHSIZE,(SP) ; ZERO IT
OFE7 1568 MOVL #<JPI$ PROCPRIV@16>+4,JPI PROC(R6) ; INITIALIZE TO GET PROCESS PRIV
OFE7 1569 MOVAB PROCPRIV(R6),JPI PROC+4(R6)
OFE7 1570 MOVL #<JPI$ IMAGPRIV@16>+4,JPI IMAG(R6) ; INITIALIZE TO GET IMAGE PRIV
OFE7 1571 MOVAB IMAGPRIV(R6),JPI IMAG+4(R6)
OFE7 1572 MOVL #<JPI$ PHDFLAGS@16>+4,JPI FLAG(R6) ; INITIALIZE TO GET PHD FLAGS
OFE7 1573 MOVAB PHD_FLAGS(R6),JPI_FLAG+4(R6)
OFE7 1574 MOVAB JPI_PROC(R6),R0 ; ADDRESS OF ITEM LIST
OFE7 1575 $GETJPI_S EFN = #EXES$C_SYSEFN,-
OFE7 1576 ITMLST = (R0)
OFE7 1577 BLBS R0,10$
OFE7 1578 5$: BRW 30$ ; ERROR - GIVE UP
OFE7 1579 10$: BBC #PHD$V_IMGDMPP,PHD_FLAGS(R6),5$ ; NO DUMP REQUESTED
OFE7 1580 BICL PROCPRIV(R6),IMAGPRIV(R6) ; TEST THAT IMAGE PRIVILEGES AREN'T GREATER
OFE7 1581 BICL PROCPRIV+4(R6),IMAGPRIV+4(R6)
OFE7 1582 BISL IMAGPRIV+4(R6),IMAGPRIV(R6)
OFE7 1583 BEQL 20$ ; NO EXCESS IMAGE PRIVILEGES
OFE7 1584 BBS #PRV$V_CHKRNLC,PROCPRIV(R6),20$
OFE7 1585 BBS #PRV$V_SETPRV,PROCPRIV(R6),20$
OFE7 1586 BRB 5$ ; INSUFFICIENT PRIVILEGES
OFE7 1587 20$: MOVL #IMGACT$ NARGS,(R6) ; SET ARGUMENT COUNT FOR $IMGACT CALL
OFE7 1588 MOVAB IMGDMPPNAM,IMGACT$ NAME(R6) ; SET ADR OF INPUT FILE NAME DESC
OFE7 1589 MOVAB DEFAULTNAMDESC,IMGACT$ DFLNAM(R6) ; SET ADR OF DEFAULT NAME STR
OFE7 1590 MOVL #<IACSM_MERGE ! IACSM-EXPREG>,IMGACT$ IMGCTL(R6) ; SET CTL FLAGS

```

```
0C A6 34 A6 9E 109D 1591      MOVAB HDRBUF(R6),IMGACT$ HDRBUF(R6) ; SET ADR OF IMAGE HEADER BUFFER
14 A6 24 A6 9E 10A2 1592      MOVAB IMGACT_INADR(R6),IMGACT$ INADR(R6) ; SET ADR OF INPUT VA RANGE
18 A6 2C A6 9E 10A7 1593      MOVAB IMGACT_RETADR(R6),IMGACT$ RETADR(R6) ; SET ADR OF RETURN RANGE
      1C A6 D4 10AC 1594      CLRL IMGACT$ IDENT(R6) ; NO MATCH IDENT SPECIFIED
24 A6 0200 8F 3C 10AF 1595      MOVZWL #^X200,IMGACT_INADR(R6) ; SET A BLUEPRINT PO ADDRESS RANGE FOR
28 A6 3FFFFFFF 8F D0 10B5 1596      MOVL #1030-1,IMGACT_INADR+4(R6) ; MAPPING TO FIRST FREE VA SPACE
      1C 50 E9 10BD 1597      $IMGACT_G (R6) ; MAP IN THE DUMP IMAGE
      10C4 1598      BLBC -R0,30$ ; ERROR - GIVE UP
      10C7 1599      $IMGFIX_S
      12 50 E9 10CE 1600      BLBC -R0,30$
51 51 2C A6 D0 10D1 1601      MOVL IMGACT_RETADR(R6),R1 ; START OF THE MERGED IN CODE
51 51 08 A1 C1 10D5 1602      ADDL3 8(R1),R1,R1 ; START ADDRESS OF THE DUMP ROUTINE
5E 0270 C6 DE 10DA 1603      MOVAL SCRATCHSIZE(R6),SP ; GET RID OF SCRATCH STORAGE
      61 16 10DF 1604      JSB (R1)
      05 11 10E1 1605      BRB 40$
5E 0270 C6 DE 10E3 1606      MOVAL SCRATCHSIZE(R6),SP ; GET RID OF SCRATCH STORAGE
      OFFC 8F BA 10E8 1607      POPR #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
      05 10EC 1608      RSB
      10ED 1609      .DSABL LSB
      10ED 1610
      10ED 1611      .SBTTL CRELNM - FIXUP AND INSERT A LOGICAL NAME BLOCK
      10ED 1612
      10ED 1613      ++
      10ED 1614      FUNCTIONAL DESCRIPTION:
      10ED 1615
      10ED 1616      THE PURPOSE OF THIS ROUTINE IS TO FIXUP A LNMB FOR A LOGICAL NAME AND INSERT
      10ED 1617      IT INTO THE APPROPRIATE HASH BUCKET OF THE PROCESS-PRIVATE LOGICAL NAME HASH
      10ED 1618      TABLE. THE LOGICAL NAME BEING FIXED REQUIRES THAT ITS EQUIVALENCE STRING BE
      10ED 1619      MOVED FROM THE PQB INTO THE STORAGE ALLOCATED FOR IT. IF THE LENGTH OF THE
      10ED 1620      EQUIVALENCE STRING IS 0 THEN THE BLOCK OF STORAGE FOR THIS LOGICAL NAME IS
      10ED 1621      DEALLOCATED AND THE ROUTINE EXITS.
      10ED 1622
      10ED 1623      CALLING SEQUENCE:
      10ED 1624
      10ED 1625      BSBW CRELNM
      10ED 1626      .WORD PQBST-<OFFSET>
      10ED 1627      .WORD PQBST-<OFFSET> ATT
      10ED 1628      .WORD <NAME>_LNMX - PROC_DIR
      10ED 1629      .WORD <NAME>- PROC_DIR
      10ED 1630
      10ED 1631      INPUT PARAMETERS:
      10ED 1632
      10ED 1633      R6 - ADDRESS OF PQB
      10ED 1634      R7 - ADDRESS OF PROCESS DIRECTORY'S TABLE HEADER
      10ED 1635      R8 - ADDRESS OF ALLOCATED P1 STORAGE
      10ED 1636      R9 - ADDRESS OF PROCESS TABLE'S TABLE HEADER
      10ED 1637
      10ED 1638      IMPLICIT INPUT:
      10ED 1639
      10ED 1640      IT IS ASSUMED THAT THE LOGICAL NAME BEING CREATED HAS ALREADY BEEN
      10ED 1641      FORMATTED WITH THE EXCEPTION OF:
      10ED 1642
      10ED 1643      1. THE CONTAINING TABLE ADDRESS WITHIN ITS LNMB.
      10ED 1644      2. THE TRANSLATION STRING AND ATTRIBUTES WITHIN ITS LNMX.
      10ED 1645
      10ED 1646      OUTPUT PARAMETERS:
      10ED 1647      NONE
```

```
10ED 1648 :  
10ED 1649 : IMPLICIT OUTPUT:  
10ED 1650 :     NONE  
10ED 1651 :  
10ED 1652 : COMPLETION CODES:  
10ED 1653 :     NONE  
10ED 1654 :  
10ED 1655 : SIDE EFFECTS:  
10ED 1656 :  
10ED 1657 :     R0 - R3, R5, AND AP ARE DESTROYED.  
10ED 1658 :  
10ED 1659 :  
10ED 1660 : CRELM:  
10ED 1661 :     MOVL      (SP),AP  
10ED 1662 :     ADDL2     #8,(SP)  
10ED 1663 :     MOVZWL    (AP)+,R1  
10ED 1664 :     MOVZBL    (R6)[R1],R0  
10ED 1665 :     BNEQ      10$,  
10ED 1666 :     MOVZWL    4(AP),R0  
10ED 1667 :     ADDL2     R8,R0  
10ED 1668 :     MOVZWL    LNMB$W_SIZE(R0),R1  
10ED 1669 :     JSB       EXE$DEAP1  
10ED 1670 :     BRB       20$  
10ED 1671 :  
10ED 1672 : 10$: MOVZWL    (AP)+,R2  
10ED 1673 :     MOVZWL    (AP)+,R3  
10ED 1674 :     MOVB      1(R6)[R2],-  
10ED 1675 :             LNMX$B_FLAGS(R8)[R3]  
10ED 1676 :     INCL      R0  
10ED 1677 :     MOVCL     R0,(R6)[R1],-  
10ED 1678 :             LNMX$T_XLATION(R8)[R3]  
10ED 1679 :     MOVZWL    (AP)+,R1  
10ED 1680 :     ADDL2     R8,R1  
10ED 1681 :     MOVL      R9,LNMB$L_TABLE(R1)  
10ED 1682 :     CLRL      R2  
10ED 1683 :     JSB       LNMB$INSLOGTAB  
10ED 1684 :     RSB  
10ED 1685 :  
10ED 1686 : 20$: RETURN
```

5C 6E DO 10ED 1661  
6E 08 CO 10ED 1662  
51 8C 3C 10ED 1663  
50 6641 9A 10ED 1664  
13 12 10ED 1665  
50 04 AC 3C 10ED 1666  
50 58 CO 10ED 1667  
51 08 A0 3C 10ED 1668  
00000000'EF 16 10ED 1669  
27 11 10ED 1670  
52 8C 3C 10ED 1672  
53 8C 3C 10ED 1673  
6843 01 A642 90 10ED 1674  
111B 1675  
111B 1676  
04 A843 6641 50 D6 10ED 1677  
2B 10ED 1678  
51 8C 3C 10ED 1679  
51 58 CO 10ED 1680  
0C A1 59 D0 10ED 1681  
52 D4 10ED 1682  
00000000'EF 16 10ED 1683  
05 10ED 1684

```
1137 1686 .SBTTL EXESCRE_JGTABLE - CREATE GROUP AND JOB-WIDE LOGICAL NAME TABLES
1137 1687 :++
1137 1688 FUNCTIONAL DESCRIPTION:
1137 1689
1137 1690 THE PURPOSE OF THIS ROUTINE IS TO HANDCRAFT GROUP AND JOB-WIDE LOGICAL
1137 1691 NAME TABLES AND DIRECT THEIR INSERTION INTO THE APPROPRIATE HASH BUCKET
1137 1692 OF THE SYSTEM LOGICAL NAME HASH TABLE. GROUP LOGICAL NAME TABLES ARE INSERTED
1137 1693 SUCH THAT IF THERE IS AN EXISTING GROUP TABLE FOR THAT GROUP, THE CALLER OF
1137 1694 THIS ROUTINE IS MAPPED TO IT.
1137 1695
1137 1696 CALLING SEQUENCE:
1137 1697
1137 1698 BSBW EXESCRE_JGTABLE
1137 1699
1137 1700 INPUT PARAMETERS:
1137 1701
1137 1702 R7 - JOB TABLE QUOTA
1137 1703 R10 - ADDRESS OF ASCII EQUIVALENT OF JIB ADDRESS
1137 1704 R11 - ADDRESS OF ASCII EQUIVALENT OF GROUP NUMBER
1137 1705
1137 1706 IMPLICIT INPUT:
1137 1707
1137 1708 LNM_SYSTEM_DIR_LNMTH - ADDRESS OF SYSTEM DIRECTORY TABLE HEADER
1137 1709 LNM$AL_HASHTBL - ADDRESS OF POINTER TO SYSTEM HASH TABLE
1137 1710
1137 1711 SCH$GL_CURPCB - ADDRESS OF PCB
1137 1712
1137 1713 OUTPUT PARAMETERS:
1137 1714 NONE
1137 1715
1137 1716 IMPLICIT OUTPUT:
1137 1717
1137 1718 R4 - ADDRESS OF PCB
1137 1719
1137 1720 COMPLETION CODES:
1137 1721
1137 1722 1 - SUCCESS
1137 1723 SSS_EXLNMQUOTA - INSUFFICIENT QUOTA IN SYSTEM DIRECTORY TABLE
1137 1724 SSS_INSMEM - INSUFFICIENT PAGED POOL TO ALLOCATE LNMB
1137 1725
1137 1726 SIDE EFFECTS:
1137 1727
1137 1728 R0 - R5 AND R8 ARE DESTROYED.
1137 1729
1137 1730 THE JOB-WIDE LOGICAL NAME TABLE WILL HAVE BEEN CREATED POTENTIALLY
1137 1731 RESULTING IN THE DELETION OF ANY SHAREABLE TABLE WITH THE SAME NAME.
1137 1732
1137 1733 THE GROUP LOGICAL NAME TABLE WILL HAVE BEEN CREATED PROVIDED A GROUP
1137 1734 TABLE WITH THAT NAME DOES NOT ALREADY EXIST IN WHICH CASE NOTHING IS
1137 1735 DONE.
1137 1736
1137 1737 :--
1137 1738
1137 1739 .ENABL LSB
1137 1740
1137 1741 EXESCRE_GTABLE::
1137 1742
```

```
1137 1743
1137 1744
1137 1745 : THIS ROUTINE EXESCRE_GTABLE IS IDENTICAL TO THE ROUTINE EXESCRE_JGTABLE
1137 1746 : WITH EXCEPTION THAT THE JOB LOGICAL NAME TABLE IS NOT CREATED. THUS THE
1137 1747 : ONLY INPUT PARAMETER IS R11, WHICH HAS THE ADDRESS OF ASCII EQUIVALENT
1137 1748 : OF GROUP NUMBER.
1137 1749
1137 1750
51 00C0 8F 3C 1137 1751 MOVZWL #GROUP TABLE SIZE,R1 ; SET SIZE OF GROUP TABLE TO BE CREATED
00000000'GF 16 1137 1752 JSB G^EXESALOPAGED ; ALLOCATE REQUIRED AMOUNT OF PAGED POOL
08 50 E8 1142 1753 BLBS R0,2$ ; CONTINUE IF ALLOCATION IS SUCCESSFUL
50 0124 8F 3C 1145 1754 MOVZWL #SS$_INSFMEM,R0 ; OTHERWISE SETUP R0 WITH ERROR CODE
0126 31 114A 1755 BRW 40$ ; AND EXIT
114D 1756
54 00000000'GF D0 114D 1757 2$: MOVL G^SCH$GL_CURPCB,R4 ; RETRIEVE PCB ADDRESS
00000000'GF 16 1154 1758 JSB G^LNMSLOCKW ; LOCK LOGICAL NAME MUTEX FOR WRITING
115A 1759
00000021'EF 51 D1 115A 1760 CMPL R1, LNMTH$L_BYTES+- ; IS THERE ENOUGH QUOTA IN THE SYSTEM
17 15 1161 1761 LNM_SYSTEM_DIR_LNMTH ; DIRECTORY TABLE?
50 52 D0 1163 1762 BLEQ 4$ ; CONTINUE IF ENOUGH QUOTA
00000000'GF 16 1166 1763 MOVL R2,R0 ; SETUP TO DEALLOCATE THE PAGED POOL
00000000'GF 16 1166 1764 JSB G^EXESDEAPGDSIZ ; DEALLOCATE IT
50 224C 8F 3C 116C 1765 JSB G^LNMSUNLOCK ; UNLOCK THE LOGICAL NAME MUTEX
00F9 31 1172 1766 MOVZWL #SS$_EXLNMQOTA,R0 ; SETUP REASON FOR PREMATURE TERMINATION
1177 1767 BRW 40$ ; AND GO RETURN TO CALLER
117A 1768
62 58 52 D0 117A 1769 4$: MOVL R2,R8 ; SAVE ADDRESS OF STORAGE ALLOCATED
F5DE CF 51 28 117D 1770 MOVCL R1, GROUP_TABLE, (R2) ; FORMAT THE LOGICAL NAME TABLE(S)
54 00000000'GF D0 1183 1771 MOVL G^SCH$GL_CURPCB,R4 ; RETRIEVE PCB ADDRESS
009C 31 118A 1772 BRW CREATE_GTABLE ; GO CREATE GROUP TABLE
118D 1773
EXESCRE_JGTABLE::
118D 1774
118D 1775
118D 1776 :
118D 1777 : ALLOCATE PAGED POOL FOR THE GROUP AND JOB-WIDE LOGICAL NAME TABLES. AFTER
118D 1778 : ALLOCATING SUFFICIENT PAGED POOL, WRITE LOCK THE LOGICAL NAME MUTEX, AND MAKE
118D 1779 : SURE THAT THE PARENT LOGICAL NAME TABLE, THE SYSTEM DIRECTORY TABLE, HAS
118D 1780 : SUFFICIENT QUOTA FOR THE CREATION OF BOTH LOGICAL NAME TABLES AND FOR ANY
118D 1781 : SEPARATE QUOTA THAT WILL BE RELEGATED TO THEM. IF THE SYSTEM DIRECTORY TABLE
118D 1782 : DOES NOT CONTAIN SUFFICIENT QUOTA THEN EXIT IMMEDIATELY WITH THE APPROPRIATE
118D 1783 : ERROR; OTHERWISE, THE BLOCK OF STORAGE THAT HAS BEEN ALLOCATED FOR THE LOGICAL
118D 1784 : NAME TABLES IS FORMATED.
118D 1785 :
118D 1786
51 0180 8F 3C 118D 1787 MOVZWL #SO_ALLOC_SIZE,R1 ; ASSUME BOTH TABLES WILL BE CREATED
00000000'GF 16 1192 1788 JSB G^EXESALOPAGED ; ALLOCATE REQUIRED AMOUNT OF PAGED POOL
07 50 E8 1198 1789 BLBS R0,1$ ; CONTINUE IF ALLOCATION IS SUCCESSFUL
50 0124 8F 3C 119B 1790 MOVZWL #SS$_INSFMEM,R0 ; OTHERWISE SETUP R0 WITH ERROR CODE
2E 11 11A0 1791 BRB 10$ ; AND EXIT
11A2 1792
54 00000000'GF D0 11A2 1793 1$: MOVL G^SCH$GL_CURPCB,R4 ; RETRIEVE PCB ADDRESS
00000000'GF 16 11A9 1794 JSB G^LNMSLOCKW ; LOCK LOGICAL NAME MUTEX FOR WRITING
11AF 1795
50 51 57 C1 11AF 1796 ADDL3 R7,R1,R0 ; DETERMINE TOTAL AMOUNT OF QUOTA
00000021'EF 50 D1 11B3 1797 CMPL R0, LNMTH$L_BYTES+- ; IS THERE ENOUGH QUOTA IN THE SYSTEM
17 15 11BA 1798 LNM_SYSTEM_DIR_LNMTH ; DIRECTORY TABLE?
11BA 1799 BLEQ 20$ ; CONTINUE IF ENOUGH QUOTA
```

```

      50 52 DO 11BC 1800      MOVL R2,R0      : SETUP TO DEALLOCATE THE PAGED POOL
      00000000'GF 16 11BF 1801      JSB G^EXESDAPGDS1Z : DEALLOCATE IT
      00000000'GF 16 11CS 1802      JSB G^LNMSUNLOCK : UNLOCK THE LOGICAL NAME MUTEX
      50 224C 8F 3C 11CB 1803      MOVZWL #SS$_EXLNMQOTA,R0 : SETUP REASON FOR PREMATURE TERMINATION
      00A0 31 11D0 1804      BRW 40$ : AND GO RETURN TO CALLER
      11D3 1805
      11D3 1806      10$: MOVL R2,R8      : SAVE ADDRESS OF STORAGE ALLOCATED
      62 58 52 DO 11D3 1806      MOVCL R1,GROUP_TABLE,(R2) : FORMAT THE LOGICAL NAME TABLE(S)
      54 F585 CF 51 28 11D6 1807      MOVL G^SCH$GL_CURPCB,R4 : RETRIEVE PCB ADDRESS
      00000000'GF DO 11DC 1808
      11E3 1809
      11E3 1810
      11E3 1811      : FIXUP THE LOGICAL NAME BLOCK FOR THE JOB TABLE THAT IS BEING CREATED, AND
      11E3 1812      : THEN INSERT IT INTO THE APPROPRIATE HASH BUCKET OF THE SHAREABLE LOGICAL NAME
      11E3 1813      : HASH TABLE. THIS FIXING UP OF THE JOB TABLE'S LOGICAL NAME BLOCK INCLUDES
      11E3 1814      : APPENDING TO THE 'LNMSJOB' ASCII STRING ALREADY PRESENT WITHIN THE NAME FIELD
      11E3 1815      : OF THE LNMB, THE ASCII EQUIVALENT OF THE JIB'S HEXADECIMAL ADDRESS. A POINTER
      11E3 1816      : TO THIS ASCII EQUIVALENCE IS PASSED TO THIS ROUTINE IN R7.
      11E3 1817
      11E3 1818
      51 00C0 C8 9E 11E3 1819      MOVAB JOB_TABLE(R8),R1 : RETRIEVE ADDRESS OF JOB TABLE'S LNMB
      1A A1 6A 7D 11E8 1820      MOVQ (R10),LNMB$NAME+9(R1) : MOVE ASCII HEX JIB ADDR INTO NAME FIELD
      00000000'FF DO 11EC 1821      MOVL @LNMS$HASH_TBL,- : MOVE THE ADDRESS OF THE SHAREABLE
      00E8 C8 11F2 1822      : JOB_TABLE_LNMB+ : LOGICAL NAME HASH TABLE INTO THE JOB
      0110 C8 9E 11F5 1823      : LNMB$HASH(R8) : TABLE'S TABLE HEADER
      00EC C8 11F9 1824      MOVAB JOB_TABLE_ORB(R8),- : MOVE THE ADDRESS OF THE JOB TABLE'S
      00F0 C8 51 DO 11FC 1825      : JOB_TABLE_LNMB+ : OBJECT RIGHTS BLOCK INTO THE JOB
      1201 1826      : LNMB$ORB(R8) : TABLE'S TABLE HEADER
      1201 1827      MOVL R1,JOB_TABLE_LNMB+ : MOVE THE ADDRESS OF THE JOB TABLE'S
      1201 1828      : LNMB$NAME(R8) : LNMB INTO THE JOB TABLE'S TABLE HEADER
      1201 1829
      00BC C4 DO 1201 1830      MOVL PCB$UIC(R4),- : MOVE THE PROCESS'S UIC INTO THE
      0110 C8 1205 1831      : JOB_TABLE_ORB+ : APPROPRIATE FIELD OF THE JOB TABLE'S
      0138 C8 7C 1208 1832      : ORB$OWNER(R8) : OBJECT RIGHTS BLOCK
      120C 1833      CLRQ JOB_TABLE_ORB+ : SET INITIAL NULL ACL
      120C 1834      : ORB$ACL_COUNT(R8)
      120C 1835
      57 52 DO 120C 1836      TSTL R7 : IS JOB TABLE QUOTA POOLED?
      11 13 120E 1837      BEQL 30$ : IF SO THEN GO INSERT LNMB
      00E7 C8 9E 1210 1838      MOVAB JOB_TABLE_LNMB(R8),- : OTHERWISE SET UP THE JOB TABLE'S
      0100 C8 1214 1839      : JOB_TABLE_LNMB+ : TABLE HEADER AS THE QUOTA HOLDER FOR
      1217 1840      : LNMB$QTABLE(R8) : THE JOB TABLE
      0104 C8 57 DO 1217 1841      MOVL R7,JOB_TABLE_LNMB+ : SET THE BYTE LIMIT FIELD TO THE
      121C 1842      : LNMB$BYTESLM(R8) : INITIAL AMOUNT OF JOB TABLE QUOTA
      0108 C8 57 DO 121C 1843      MOVL R7,JOB_TABLE_LNMB+ : SET THE BYTE REMAINING FIELD TO THE
      1221 1844      : LNMB$BYTES(R8) : INITIAL AMOUNT OF JOB TABLE QUOTA
      52 D4 1221 1845      30$: CLRL R2 : NO SPECIAL INSERTION ATTRIBUTES
      00000000'GF 16 1223 1846      JSB G^LNMSINSLOGTAB : APPROPRIATELY INSERT LNMSGROUP_XXXXXX
      1229 1847
      1229 1848
      1229 1849      : FIXUP THE LOGICAL NAME BLOCK FOR THE GROUP TABLE THAT IS BEING CREATED, AND
      1229 1850      : THEN INSERT IT INTO THE APPROPRIATE HASH BUCKET OF THE SHAREABLE LOGICAL NAME
      1229 1851      : HASH TABLE PROVIDED A TABLE FOR THAT GROUP DOES NOT ALREADY EXIST. THIS
      1229 1852      : FIXING UP OF THE GROUP TABLE'S LOGICAL NAME BLOCK INCLUDES APPENDING TO THE
      1229 1853      : 'LNMSGROUP' ASCII STRING ALREADY PRESENT WITHIN THE NAME FIELD OF THE LNMB,
      1229 1854      : THE ASCII EQUIVALENT OF THE 'OCTAL GROUP' THE PROCESS BELONGS TO. A POINTER
      1229 1855      : TO THIS ASCII EQUIVALENCE IS PASSED TO THIS ROUTINE IN R8.
      1229 1856
```

```

      1229 1857
      1229 1858 CREATE_GTABLE:
      1229 1859      MOVL R8,R1      : SETUP TO INSERT THE GROUP TABLE'S LNMB
      122C 1860      MOVL (R1),LNMB$T_NAME+11(R1) : APPEND THE ASCII 'OCTAL GROUP' TO THE
      1230 1861      MOVW 4(R1),LNMB$T_NAME+15(R1) : 'LNMSGROUP' ALREADY IN THE NAME FIELD
      1235 1862      MOVL @LNMB$AL_HASHTBL,-      : MOVE THE ADDRESS OF THE SHAREABLE
      1238 1863      GROUP_TABLE_LNMTH+-      : LOGICAL NAME HASH TABLE INTO THE GROUP
      123D 1864      LNMB$SL_HASH(R8)      : LOGICAL NAME TABLE'S YABLE HEADER
      123D 1865      MOVAB GROUP_TABLE_ORB(R8),-      : MOVE THE ADDRESS OF THE GROUP TABLE'S
      1240 1866      GROUP_TABLE_LNMTH+-      : OBJECT RIGHTS BLOCK INTO THE GROUP
      1242 1867      LNMB$SL_ORB(R8)      : TABLE'S TABLE HEADER
      1242 1868      MOVL R1,GROUP_TABLE_LNMTH+-      : MOVE THE ADDRESS OF THE LNMB INTO THE
      1246 1869      LNMB$SL_NAME(R8)      : GROUP TABLE'S TABLE HEADER
      1246 1870
      1246 1871      MOVW PCBSW_GRP(R4),-      : MOVE THE PROCESS'S GROUP NUMBER
      124A 1872      GROUP_TABLE_ORB+-      : INTO THE APPROPRIATE FIELD OF THE
      124C 1873      ORB$SL_OWNER+2(R8)      : GROUP TABLE'S OBJECT RIGHTS BLOCK
      124C 1874      CLRQ GROUP_TABLE_ORB+-      : SET INITIAL NULL ACL
      124F 1875      ORB$SL_ACL_COUNT(R8)
      124F 1876
      124F 1877      MOVL #LNMB$M_CREATE_IF,R2      : GROUP TABLES ARE INSERTED CREATE_IF
      1256 1878      JSB G*LNMB$INSLOGTAB      : APPROPRIATELY INSERT LNMSGROUP_XXXXXX
      125C 1879
      125C 1880      CMPW #SS$ _NORMAL,R0      : DID THE GROUP TABLE ALREADY EXIST?
      125F 1881      BNEQ 35$      : GO UNLOCK THE MUTEX IF IT DIDN'T
      1261 1882      MOVL R8,R0      : DELETE THE LNMB FOR WHAT WOULD HAVE
      1264 1883      JSB G*EX$DEAPAGED      : BECOME A GROUP LOGICAL NAME TABLE
      126A 1884      :
      126A 1885      : UNLOCK THE LOGICAL NAME MUTEX AND RETURN STATUS.
      126A 1886      :
      126A 1887
      126A 1888      35$: JSB G*LNMB$UNLOCK      : UNLOCK THE LOGICAL NAME MUTEX
      1270 1889      MOVZBL #1,R0      : SUCCESS
      1273 1890      40$: RSB      : RETURN
      1274 1891
      1274 1892      .DSABL LSB
      1274 1893
      1274 1894      .END

```

Variable	Value	Access	Mode
SSARGS	= 00000008		
SS1	= 00000001		
CCBSC_LENGTH	= 00000010		
CHARS	0000001C	R	02
CHFSL_MCHARGLST	= 00000008		
CHFSL_MCH_DEPTH	= 00000008		
CHFSL_SIGARGLST	= 00000004		
CHFSL_SIG_NAME	= 00000004		
CLIS_INVREQTYP	= 00038822		
CREATE_GTABLE	00001229	R	02
CRELNM	000010ED	R	02
CTLSAL_CLICALBK	*****	X	02
CTLSA_DISPVEC	*****	X	02
CTLSC_KRP_COUNT	*****	X	02
CTLSC_KRP_SIZE	*****	X	02
CTL\$GB_MSGMASK	*****	X	02
CTL\$GB_SSFILTER	*****	X	02
CTL\$GL_CCBASE	*****	X	02
CTL\$GL_CREPRC_FLAGS	*****	X	02
CTL\$GL_CTLBASVA	*****	X	02
CTL\$GL_GETMSG	*****	X	02
CTL\$GL_IAFLAST	*****	X	02
CTL\$GL_IAFLINK	*****	X	02
CTL\$GL_IAFPERM	*****	X	02
CTL\$GL_KRP	*****	X	02
CTL\$GL_KRPFL	*****	X	02
CTL\$GL_LNMDIRECT	*****	X	02
CTL\$GL_LNMHASH	*****	X	02
CTL\$GL_PCB	*****	X	02
CTL\$GL_PHD	*****	X	02
CTL\$GL_PRCALLCNT	*****	X	02
CTL\$GL_RMSBASE	*****	X	02
CTL\$GL_UAF_FLAGS	*****	X	02
CTL\$GL_USRCHME	*****	X	02
CTL\$GL_USRCHMK	*****	X	02
CTL\$GL_USRUNDWN	*****	X	02
CTL\$GQ_ALLOCREG	*****	X	02
CTL\$GQ_LNMTBLCACHE	*****	X	02
CTL\$GQ_LOGIN	*****	X	02
CTL\$GQ_PROCPRIV	*****	X	02
CTL\$GT_CLINAME	*****	X	02
CTL\$GW_NMIOCH	*****	X	02
CTL\$T_USERNAME	*****	X	02
DEFAULTNAMDSC	0000003F	R	02
DEFDESC	00000010	R	02
DIR...	= 00000001		
DIVR	00000AE2	R	02
DYN\$C_LNM	= 00000040		
DYN\$C_ORB	= 00000049		
DYN\$C_RSHT	= 00000038		
EXESALOP1PROC	*****	X	02
EXESALOPAGED	*****	X	02
EXESATCH_ALL	00000E80	RG	02
EXESCLI_UTILSRV	00000E69	RG	02
EXESCRE_GTABLE	00001137	RG	02
EXESCRE_JGTABLE	0000118D	RG	02
EXESC_SYSEFN	*****	X	02

NAME	VALUE	UNIT	TYPE	MODE
EXESDEAP1	*****		X	02
EXESDEAPAGED	*****		X	02
EXESDEAPGDSIZ	*****		X	02
EXESEXCEPTABLE	*****		X	02
EXESEXCMMSG	*****		X	02
EXESEXIT_IMAGE	00000E73	RG		02
EXESGL_FLAGS	*****		X	02
EXESGL_POBBL	*****		X	02
EXESGQ_SYSDISK	00000000	RG		02
EXESGQ_SYSTIME	*****		X	02
EXESIMGDMP_EXEC	00000FE7	RG		02
EXESIMGDMP_MERGE	00000FED	RG		02
EXESPROCIMACT	00000D88	RG		02
EXESPROCSTRT	00000000	RG		03
EXESRMSEXH	00000F25	RG		02
EXESV_INIT	*****		X	02
EXEC_A	00000FFD	R		02
EXE_PROCSTRT	000008E0	R		02
GROUP	= 000000E0			
GROUP_SIZE	= 00000038			
GROUP_TABLE	= 00000760	R		02
GROUP_TABLE_LNMTH	= 00000027			
GROUP_TABLE_ORB	= 00000050			
GROUP_TABLE_ORB_SIZE	= 00000070			
GROUP_TABLE_SIZE	= 000000C0			
GROUP_XEND_SIZE	= 00000031			
HDRBUF	00000034			
IAC\$AW_VECRESET	*****		X	02
IAC\$AW_VECSET	*****		X	02
IAC\$GL_ICBFL	*****		X	02
IAC\$GL_IMAGE_LIST	*****		X	02
IAC\$GL_WORK_LIST	*****		X	02
IAC\$M_EXPREG	= 00000020			
IAC\$M_MERGE	= 00000010			
IHD\$SL_LNKFLAGS	= 00000020			
IHD\$W_ACTIVOFF	= 00000002			
IMAGPRIV	0000023C			
IMGACT\$_ACMODE	= 00000020			
IMGACT\$_DFLNAM	= 00000008			
IMGACT\$_HDRBUF	= 0000000C			
IMGACT\$_IDENT	= 0000001C			
IMGACT\$_IMGCTL	= 00000010			
IMGACT\$_INADR	= 00000014			
IMGACT\$_NAME	= 00000004			
IMGACT\$_NARGS	= 00000008			
IMGACT\$_RETADR	= 00000018			
IMGACT_INADR	00000024			
IMGACT_RETADR	0000002C			
IMGDMPNAM	00000057	R		02
IMGNAM	00000CEE	R		02
IMPSL_IOSEGADDR	= 00000004			
JIB\$S_ACCOUNT	= 00000008			
JIB\$S_USERNAME	= 0000000C			
JIB\$T_ACCOUNT	= 00000018			
JIB\$T_USERNAME	= 0000000C			
JOB	= 00000118			
JOB_SIZE	= 00000030			

PROCSTR  
Symbol table

I 12  
- PROCESS STARTUP AND INITIALIZATION

16-SEP-1984 01:00:43 VAX/VMS Macro V04-00  
14-SEP-1984 22:32:32 [SYS.SRC]PROCSTR.MAR;3

Page 42  
(11)

```

JOB_TABLE = 000000C0
JOB_TABLE_LNMTH = 000000E7
JOB_TABLE_ORB = 00000110
JOB_TABLE_ORB_SIZE = 00000070
JOB_TABLE_SIZE = 000000C0
JOB_XEND_SIZE = 0000002F
JPI$-IMAGPRIV = 00000413
JPI$-PHD$FLAGS = 0000041B
JPI$-PROCPRIV = 00000204
JPI_END = 0000026C
JPI_FLAG = 00000260
JPI_IMAG = 00000254
JPI_PROC = 00000248
LNMB$AL_HASHTBL ***** X 02
LNMB$GL_HTBL$SIZE ***** X 02
LNMB$HASH ***** X 02
LNMB$INSLOGTAB ***** X 02
LNMB$LOCKW ***** X 02
LNMB$M_CREATE_IF = 01000000 ***** X 02
LNMB$B_ACMODE = 0000000B
LNMB$B_FLAGS = 00000010
LNMB$B_TYPE = 0000000A
LNMB$B_LINK = 00000004
LNMB$B_LINK = 00000000
LNMB$B_TABLE = 0000000C
LNMB$M_NODELETE = 00000010
LNMB$M_NO_ALIAS = 00000001
LNMB$M_TABLE = 0000000B
LNMB$T_NAME = 00000011
LNMB$W_SIZE = 0000000B
LNMB$K_LENGTH = 00000080
LNMB$K_TBLADDR = 0000000C
LNMB$W_SIZE = 0000000B
LNMB$H$B_TYPE = 0000000A
LNMB$H$B_BUCKET = 0000000C
LNMB$H$B_BUCKET = 0000000C
LNMB$H$B_MASK = 00000000
LNMB$H$B_SIZE = 0000000B
LNMB$H$B_FLAGS = 00000000
LNMB$H$B_LENGTH = 00000025
LNMB$H$B_BYTES = 00000021
LNMB$H$B_BYTESLM = 0000001D
LNMB$H$B_CHILD = 00000011
LNMB$H$B_HASH = 00000001
LNMB$H$B_NAME = 00000009
LNMB$H$B_ORB = 00000005
LNMB$H$B_PARENT = 0000000D
LNMB$H$B_QTABLE = 00000019
LNMB$H$B_SIBLING = 00000015
LNMB$H$B_DIRECTORY = 00000002
LNMB$H$B_GROUP = 00000004
LNMB$H$B_SHAREABLE = 00000001
LNMB$B_FLAGS = 00000000
LNMB$B_INDEX = 00000001
LNMB$K_TABLE = FFFFFFFF82
LNMB$M_TERMINAL = 00000002

```

```

LNMB$M_XEND = 00000004
LNMB$T_XLATION = 00000004
LNMB$W_HASH = 00000002
LNMB$SYSTEM_DIR_LNMTH ***** X 02
LNMB$SECRETVA ***** X 02
LNMB$GL_CTLBASVA ***** X 02
LNMB$GL_RMSBASE ***** X 02
LNMB$IMGHDRBUF ***** X 02
LNMB$IMGRESET ***** X 02
NXT$EVEC = 00000100
NXT$KVEC = 00000000
NXT$MVEC = 00000300
NXT$RVEC = 00000200
OPS_RSB = 00000005
ORB$B_FLAGS = 0000000B
ORB$B_TYPE = 0000000A
ORB$K_LENGTH = 0000005B
ORB$K_ACL_COUNT = 0000002B
ORB$K_ACL_DESC = 0000002C
ORB$K_ACL_MUTEX = 00000004
ORB$K_GRP_PROT = 00000020
ORB$K_OWNER = 00000000
ORB$K_OWN_PROT = 0000001C
ORB$K_SYS_PROT = 0000001B
ORB$K_WOR_PROT = 00000024
ORB$Q_MODE_PROT = 00000010
ORB$R_MAX_CLASS = 00000044
ORB$R_MIN_CLASS = 00000030
ORB$S_MAX_CLASS = 00000014
ORB$S_MIN_CLASS = 00000014
ORB$W_REF_COUNT = 0000000E
ORB$W_SIZE = 0000000B
P1_ALLOC_SIZE = 000006F0
PCB$B_AUTHPRI = 0000002B
PCB$B_PRI = 0000002F
PCB$K_JIB = 00000080
PCB$K_OWNER = 0000001C
PCB$K_PHD = 0000006C
PCB$K_PQB = 0000004C
PCB$K_STS = 00000024
PCB$K_UIC = 000000BC
PCB$V_HIBER = 00000013
PCB$W_GRP = 000000BE ***** X 02
PFN$GL_PHYPGCNT = 0000010C
PHD$B_AUTHPRI = 0000005C
PHD$K_CPU_LIM = 00000020
PHD$M_IMGDM = 000000E0
PHD$Q_AUTHPRIV = 00000000
PHD$Q_PRIVMSK = 0000012B
PHD$R_MAX_CLASS = 00000114
PHD$R_MIN_CLASS = 00000014
PHD$S_MAX_CLASS = 00000014
PHD$S_MIN_CLASS = 00000005
PHD$V_IMGDM = 00000040
PHD$W_DFWSCNT = 0000001A
PHD$W_FLAGS = 00000036

```

PROCSTRT  
Symbol table

J 12  
- PROCESS STARTUP AND INITIALIZATION

16-SEP-1984 01:00:43 VAX/VMS Macro V04-00  
14-SEP-1984 22:32:32 [SYS.SRC]PROCSTRT.MAR;3

Page 43  
(11)

PHDSW_WSAUTH	= 0000000A		
PHDSW_WSAUTHEXT	= 00000014		
PHDSW_WSEXTENT	= 00000016		
PHDSW_WSLIST	= 00000008		
PHDSW_WSQUOTA	= 00000018		
PHD_FLAGS	00000244		
PIOSAL_RMSEXH	*****	X	02
PIOSGQ_IIODEFAULT	*****	X	02
PIOSGT_DDSTRING	*****	X	02
PIOSGW_PIOIMPA	*****	X	02
PQBSB_MSGMASK	= 00000046		
PQBSL_ASTLM	= 0000000C		
PQBSL_CPULM	= 00000018		
PQBSL_CREPRC_FLAGS	= 0000004C		
PQBSL_DISK_ATT	= 00000084		
PQBSL_ERROR_ATT	= 00000080		
PQBSL_INPUT_ATT	= 00000078		
PQBSL_JTQUOTA	= 00000040		
PQBSL_OUTPUT_ATT	= 0000007C		
PQBSL_UAF_FLAGS	= 00000048		
PQBSL_WSDEFAULT	= 00000034		
PQBSL_WSEXTENT	= 0000003C		
PQBSL_WSQUOTA	= 00000030		
PQBSQ_PRVMSK	= 00000000		
PQBSR_MAX_CLASS	= 00000064		
PQBSR_MIN_CLASS	= 00000050		
PQBS\$CLI_NAME	= 00000020		
PQBS\$CLI_TABLE	= 00000100		
PQBS\$DDSTRING	= 00000100		
PQBS\$DISK	= 00000100		
PQBS\$ERROR	= 00000100		
PQBS\$INPUT	= 00000100		
PQBS\$MAX_CLASS	= 00000014		
PQBS\$MIN_CLASS	= 00000014		
PQBS\$OUTPUT	= 00000100		
PQBS\$SPAWN_CLI	= 00000020		
PQBS\$SPAWN_TABLE	= 00000100		
PQBST_CLI_NAME	= 00000088		
PQBST_CLI_TABLE	= 000000A8		
PQBST_DDSTRING	= 00000068		
PQBST_DISK	= 000005C8		
PQBST_ERROR	= 000004C8		
PQBST_IMAGE	= 000007C8		
PQBST_INPUT	= 000002C8		
PQBST_OUTPUT	= 000003C8		
PQBST_SPAWN_CLI	= 000001A8		
PQBST_SPAWN_TABLE	= 000001C8		
PQBSV_IMGDMF	= 00000000		
PQBSW_FLAGS	= 00000044		
PR\$ IPL	= 00000012		
PROCESS	= 000000A8		
PROCESS_SIZE	= 00000038		
PROCPRI\$	00000234		
PROC_DIR	00000068	R	02
PROC_DIR_LNMTH	= 0000002C		
PROC_DIR_SIZE	= 00000058		
PROC_TABLE	= 00000058		

PROC_TABLE_LNMTH	= 00000080		
PROC_TABLE_SIZE	= 00000050		
PRT\$C_UREW	= 00000000		
PRV\$V_CMKRN	= 00000000		
PRV\$V_SETPRV	= 0000000E		
PSL\$C_EXEC	= 00000001		
PSL\$C_KERNEL	= 00000000		
PSL\$C_USER	= 00000003		
PSL\$S_CURMOD	= 00000002		
PSL\$V_CURMOD	= 00000018		
PSL\$V_PRVMOD	= 00000016		
RMS\$ BUSY	= 0001848C		
SO ACLOC_SIZE	= 00000180		
SAVABS...	= 00000270		
SCH\$GL_CURPCB	*****	X	03
SCH\$GL_FREELIM	*****	X	02
SCRATCHSIZE	00000270		
SEC\$M_EXPREG	= 00020000		
SEC\$M_SYSGBL	= 00008000		
SGN\$GC_MAXWSCNT	*****	X	02
SGN\$GW_CTLIMGLIM	*****	X	02
SGN\$GW_CTLPAGES	*****	X	02
SGN\$GW_IMGIOCNT	*****	X	02
SGN\$GW_PCHANCNT	*****	X	02
SGN\$GW_PIOFAGES	*****	X	02
SS\$ CONTINUE	= 00000001		
SS\$ EXLNMQUOTA	= 0000224C		
SS\$ IN\$FME	= 00000124		
SS\$ NORMAL	= 00000001		
SS\$ SSFAIL	= 0000045C		
ST\$K_SEVERE	= 00000004		
ST\$S\$ CODE	= 0000000C		
ST\$S\$ SEVERITY	= 00000003		
ST\$V CODE	= 00000003		
ST\$V INHIB MSG	= 0000001C		
ST\$V SEVERITY	= 00000000		
SUFFIX	0000002C	R	02
SY\$CMKRN	*****	GX	02
SY\$DCLEXH	*****	GX	02
SY\$DISK	= 000005D0		
SY\$DISK_LNM	= 000005EA		
SY\$DISK_SIZE	= 00000120		
SY\$ERROR	= 000004B0		
SY\$ERROR_LNM	= 000004CB		
SY\$ERROR_SIZE	= 00000120		
SY\$EXIT	*****	X	02
SY\$EXPREG	*****	GX	02
SY\$GETJPI	*****	GX	02
SY\$HIBER	*****	GX	02
SY\$IMGACT	*****	GX	02
SY\$IMGFIX	*****	GX	02
SY\$INPUT	= 00000148		
SY\$INPUT_LNM	= 00000163		
SY\$INPUT_SIZE	= 00000120		
SY\$MGBLSC	*****	GX	02
SY\$OUTPUT	= 00000388		
SY\$OUTPUT_LNM	= 000003A4		

PROCSTRT  
Symbol table

- PROCESS STARTUP AND INITIALIZATION<sup>K 12</sup>

16-SEP-1984 01:00:43 VAX/VMS Macro V04-00  
14-SEP-1984 22:32:32 [SYS.SRC]PROCSTRT.MAR;3

Page 44  
(11)

SYSS\$OUTPUT\_SIZE = 00000128  
SYSS\$PUTMSG \*\*\*\*\* X 02  
SYSS\$RMSRUNDWN \*\*\*\*\* X 02  
SYSS\$SETEXV \*\*\*\*\* GX 02  
SYSS\$SETSFM \*\*\*\*\* GX 02  
TT = 00000268  
TT\_LNMX = 0000027C  
TT\_SIZE = 00000120  
VABUG 00000AD5 R 02  
XQP\$GL\_DZRO \*\*\*\*\* X 02  
XQP\$GL\_SECTIONS \*\*\*\*\* X 02  
XQPMERGE 00000F4C R 02  
XQP\_NAM 00000FDD R 02  
XQP\_NAM\$IZ = 0000000A

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000270 ( 624.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
YYPROCSTRT	00001274 ( 4724.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC 21
AEXENONPAGED	00000011 ( 17.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.06	00:00:00.70
Command processing	112	00:00:00.50	00:00:03.78
Pass 1	601	00:00:27.11	00:01:35.27
Symbol table sort	0	00:00:03.87	00:00:10.29
Pass 2	363	00:00:06.72	00:00:23.84
Symbol table output	42	00:00:00.31	00:00:01.35
Psect synopsis output	2	00:00:00.03	00:00:00.18
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1152	00:00:38.61	00:02:15.41

The working set limit was 2250 pages.  
154901 bytes (303 pages) of virtual memory were used to buffer the intermediate code.  
There were 130 pages of symbol table space allocated to hold 2571 non-local and 54 local symbols.  
1894 source lines were read in Pass 1, producing 37 object records in Pass 2.  
54 pages of virtual memory were used to define 52 macros.

-----  
! Macro library statistics !  
-----

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	12
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	36
TOTALS (all libraries)	48

2738 GETS were required to define 48 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:PROCSTRT/OBJ=OBJ\$:PROCSTRT MSRC\$:PROCSTRT/UPDATE=(ENH\$:PROCSTRT)+EXECMLS/LIB

0379 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY